LIVERPOOL JOHN MOORES UNIVERSITY

LJMU Research Online

Naureen, A, Zhang, N, Furber, S and Shi, Q

A GPS-Less Localization and Mobility Modelling (LMM) System for Wildlife Tracking

http://researchonline.ljmu.ac.uk/id/eprint/13238/

Article

Citation (please note it is advisable to refer to the publisher's version if you intend to cite from this work)

Naureen, A, Zhang, N, Furber, S and Shi, Q (2020) A GPS-Less Localization and Mobility Modelling (LMM) System for Wildlife Tracking. IEEE Access, 8. 102709 -102732. ISSN 2169-3536

LJMU has developed **LJMU Research Online** for users to access the research output of the University more effectively. Copyright © and Moral Rights for the papers on this site are retained by the individual authors and/or other copyright owners. Users may download and/or print one copy of any article(s) in LJMU Research Online to facilitate their private study or for non-commercial research. You may not engage in further distribution of the material or use it for any profit-making activities or any commercial gain.

The version presented here may differ from the published version or from the version of the record. Please see the repository URL above for details on accessing the published version and note that access may require a subscription.

For more information please contact researchonline@ljmu.ac.uk

http://researchonline.ljmu.ac.uk/

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 12, 2020, accepted May 20, 2020, date of publication May 26, 2020, date of current version June 11, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2997723

A GPS-Less Localization and Mobility Modelling (LMM) System for Wildlife Tracking

AYESHA NAUREEN[1], NING ZHANG[1], STEVE FURBER[2], (Fellow, IEEE), AND QI SHI[3]

[1]Information Management Research Group, Department of Computer Science, The University of Manchester, Manchester M13 9PL, U.K.
[2]Advanced Processor Technology Research Group, Department of Computer Science, The University of Manchester, Manchester M13 9PL, U.K.
[3]Department of Computer Science, Liverpool John Moores University, Liverpool L3 3AF, U.K.

Corresponding author: Ayesha Naureen (ayesha.naureen@manchester.ac.uk)

ABSTRACT Existing wildlife tracking solutions typically use sensor nodes with specialised facilities, such as long-range radio, solar array of cells and Global Positioning System (GPS). This introduces additional manufacturing cost, increased energy and memory consumptions and increased sensor node weight. This paper proposes a novel Localization and Mobility Modelling (LMM) system, that can carry out wildlife tracking by merely using low-cost, lightweight sensor nodes and using short-range peer-to-peer communication facilities only, i.e. without the need for any specialised facilities. This is done by using two computationally simple operations, which are: (i) aggregated data collections from sensor nodes via peer-to-peer communications in a distributed manner, and (ii) estimation of sensor nodes' movement traces using trilateration. The computational load placed on each sensor node is just that of data collection and aggregation, whereas movement traces estimation is carried out on a backend server, separated from the sensor nodes. In the design of the LMM system, we have: (i) carried out an empirical evaluation of different parameter value settings for data collection to develop a Multi-Zone Multi-Hierarchy (MZMH) communication structure, (ii) demonstrated a novel use of an Aggregation based Topology Learning (ATL) protocol for collecting sensor nodes' topology data using peer-to-peer multi-hop communications, and (iii) used a novel Location Estimation (LE) method for estimating sensor nodes' movement traces from the collected topology data. The evaluation results show that the LMM system can accurately estimate sensor nodes' movement traces but with significantly less energy and memory costs, demonstrating its cost-efficiency as compared to the related wildlife tracking solutions.

INDEX TERMS Internet of Things, peer to peer computing, telemetry, wireless sensor networks.

I. INTRODUCTION

Radio telemetry has been used for wildlife tracking since the 1960s. In radio telemetry, a radio transmitter is attached to an animal (e.g. as a tracking collar) and the signals transmitted by the radio transmitter are used to locate and track the animal. Over the years, many Wireless Sensor Network (WSN) based radio telemetry systems have been proposed [1]–[9]. Such a system typically consists of a set of sensor nodes, each representing a tracking collar worn by an animal, and a base station (BS), located on a driven-by (or flown-over) vehicle. In addition to a short-range radio and a microcontroller, each sensor node is also equipped with a Global Positioning System (GPS) facility, a long-range radio and some solar cells. The GPS facility determines the physical location of the node (i.e. the animal), the long-range radio periodically transmits the location data to the BS and the solar cells provides power for the node's operation. These components, i.e. the GPS, long-range radio and solar cells increase the weight of each tracking collar, the deployment cost of the system and also increase the energy and memory consumptions at the sensor node.

The weight of a tracking collar placed on an animal can significantly affect the movement of the animal. To reduce such an effect, it has been suggested that an acceptable maximum weight of a tracking collar should be less than 5% of the body mass of an animal that wears the collar [10]–[12]. Some studies, e.g. [13], [14], have also shown that, even within this acceptable range, a very small variation in the tracking collar weight and fit can still have a significant effect on the animal's movement pattern. This means that the movement data collected from animals wearing heavy tracking collars

The associate editor coordinating the review of this manuscript and approving it for publication was Honghao Gao.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

may not accurately capture the normal movement pattern of the animals. To alleviate the problem, the tracking collar weight should be reduced as much as possible.

One way to reduce the collar weight is to replace the GPS-equipped sensor nodes with non-GPS (or ordinary) sensor nodes. This means that there should be a mechanism to learn (or derive) the location of each sensor node and to transfer such location data to the BS via short-range radio communications. Such an ordinary sensor node based animal tracking system can not only reduce the weight of each sensor node, but also impose a lower level of energy requirement, as short-range communications generally cost less power than long-range communications. A lower level of energy requirement leads to a lower battery capacity requirement, or given the same battery capacity, a longer sensor node lifetime.

This paper reports the design, implementation and evaluation of an ordinary sensor node based animal tracking system, called a Localization and Mobility Modelling (LMM) system. Each wildlife tracking collar is an ordinary sensor node, which typically consists of a wireless transceiver, an on-board antenna, a microcontroller and 2 x AA cells. Such a sensor node can only communicate within a short radio range (e.g. 50 - 125 m for Tmote Sky [15]) and store and execute lightweight code (e.g. on an 8MHz microcontroller with 10k RAM and 48k Flash for Tmote Sky).

Major challenges in designing such a system is how to learn animal movement traces, i.e. how to learn the locations of moving sensor nodes continuously, and do so with resource-constrained sensor nodes, which have limited computing and storage resources and are only equipped with short-range radio communication facilities.

To address the challenges, we have used a number of ideas. The first idea is to use a Multi-Zone Multi-Hierarchical (MZMH) system architecture and communication structure to allow parallelism in communications. The architecture consists of a Backend Server, a mobile BS, multiple stationary BSes and a large number of mobile sensor nodes (each sensor node is a tracking collar worn by an animal). The whole geographical area covered by the mobile sensor nodes is divided into multiple zones and each zone is equipped with a stationary BS. Each stationary BS is responsible for collecting the topology data of the sensor nodes in the zone, in terms of its hop-wise position with respect to the stationary BS. The mobile BS periodically travels to each zone and collects the data from each stationary BS in the network. The mobile BS then forwards the collected data to the Backend Server, which estimates the movement traces of the sensor nodes based on the data received over time. From this structure, we can see that the sensor nodes' location data in different zones are collected in parallel. Additionally, the data collections from the stationary BSes by the mobile BS are also pipelined with the data collections from the sensor nodes by the stationary BSes in each zone. This parallelization and pipelining in data collection operations can shorten the total time required to complete each round of data collections.

The second idea is to use a Data Collection with Encoding and Aggregation Support (DCwEAS) approach to collect the topology data in each zone. Each zone can potentially host a large number of sensor nodes. Collecting topology data from a large number of mobile sensor nodes with limited storage capacity and limited short-range communication capability is a challenging task. This is because, to track the traces of mobile nodes (i.e. the moving animals), the topology data should be collected periodically with a sufficiently high frequency. This indicates that each data collection from all the nodes in the network should be accomplished within a sufficiently short interval, and this would not be possible with a centralised data collection method, such as a round-robin based method by which the stationary BS collects data sequentially from each of the nodes in a zone. In addition, some nodes may not be within the radio range of the stationary BS, thus requiring data being forwarded by other nodes nearer to the stationary BS. However, data forwarding is a very power consuming operation and the amount of power consumed by each forwarding node is dependent on the number of messages it forwards and the length of each such message – the fewer the messages and the shorter the messages that each node needs to transmit, the less power it consumes. In addition, each sensor node only has a very small storage capacity, so the length of the messages transmitted is very small too (e.g. a standard TinyOS message is just 36 bytes long with 5 bytes of header, 29 bytes of payload and 2 bytes of CRC [16]). To accommodate these constraints, data aggregation should be used, i.e. each node should aggregate the data that it receives (as a forwarding node) along with the data that it generates before transmitting the data to its upstream neighbour towards the stationary BS in the zone. The data aggregation should preserve the topological information contained in the data and the preservation is done by using a special data structure that encodes the topology information. This DCwEAS idea has been implemented in the Aggregation based Topology Learning (ATL) protocol that is used to accomplish the data collections in each zone. Basically, the sensor nodes in a zone form a tree topology with the stationary BS as the treehead. The sensor nodes' topology data, in terms of their respective hop distances from the treehead (i.e. the number of hops between the nodes and the treehead), are collected hop-by-hop in a bottom-up fashion, i.e. each transmission of such data starts from the leaf nodes. At each node, when data are received, they are aggregated along with local data and then transmitted to the upstream nodes. This process continues until the treehead is reached. A special data structure based data encoding method is used to encode the sensor nodes' topology data, so that the data encoding method, combined with the measure of data aggregation, reduces the size of the data that are carried in each message being transmitted. In fact, with our data encoding and data aggregation method, we make the length of the data carried in each message independent of the amount of data collected.

The third idea is for estimating the movement traces of sensor nodes in the network based on their topology data that

are collected periodically and in real-time. For this, we have devised a range-free Location Estimation (LE) method, that is a variation of the DV-hop algorithm [17]. The method is based on trilateration.[1] Given the physical locations of at least three stationary BSes and the hop distances of a sensor node from these BSes, the LE method can estimate the sensor node's physical location at a given time and the estimation of the node's locations over time produces the movement trace of the sensor node. The physical locations of all the stationary BSes in the network are already known by the Backend Server and the hop distances of the sensor nodes measured from the stationary BSes are the nodes' topology data mentioned above.

To implement the above mentioned ideas, thus facilitating the operation of the LMM system, we have designed and implemented three protocols, in addition to the LE method. The three protocols are the Aggregation based Topology Learning (ATL) protocol, MDC (Master Data Collector) Data Collection protocol and MDC Data Transmission protocol. These protocols are, respectively, for sensor nodes' topology data collections in a particular zone covered by a stationary BS, for topology data collections from the stationary BSes by the mobile BS and for transmission of the collected data by the mobile BS to the Backend Server.

The effectiveness (in terms of the amount of the topology data collected and the accuracy of the estimated movement traces) and the efficiency (in terms of the node weight, node storage capacity, energy cost and deployment cost) of the LMM system have been evaluated using simulation studies and real movement traces of zebras [18] are used in these studies. The simulation results have been compared with the results from related work.

It should be emphasized that, although the research reported in this paper is in the context of wildlife tracking, the designed system and research findings can also be applied to other application areas, such as Internet of Things (IoT) and edge computing. For example, in Vehicle-to-Vehicle (V2V) communications [19], which is one of the IoT applications, the functions running on our sensor nodes can be part of the on-board system on each vehicle and the functions running on BSes can be running on road side devices. In the edge computing context [20], the sensor node functions can be installed on end user devices and the BS functions can be running on access network devices.

The remainder of this paper is organized as follows. The related work is discussed in Section II. Section III presents the design preliminaries of LMM, namely notations, assumptions and design requirements. Section IV gives an overview of the LMM system before describing it in detail in Section V. Section VI analyses the performance of the LMM system using simulation studies, while Section VII presents a comparison of the LMM system with related work. Finally, Section VIII concludes the paper.

[1] A range-free localization method that determines the location of an object in a 2D plane using its distances from at least three reference points, with known locations. With the distances and the known locations, the object's location is calculated as an intersection point on a set of triangles.

II. RELATED WORK

This section gives an overview of related solutions so as to highlight the need for the work presented in this paper. The solutions can largely be classified into two groups: (A) those carried out to apply WSN technologies to wildlife tracking, and (B) those carried out to identify and study the effects of wildlife tracking devices on the movements and behaviour of animals.

A. WSN BASED WILDLIFE TRACKING SOLUTIONS

Existing WSN based wildlife tracking solutions [1]–[9] are proposed for tracking different types of animals, such as cows, horses, sheep, birds, iguanas and zebras. These different types of animals differ in terms of their habitat, speed, movement patterns etc., thus the solutions designed for them are also different. The solutions that are most relevant to ours are the solutions reported in [6]–[9].

Juang *et al.* [6] propose a peer-to-peer WSN, called ZebraNet, for zebra tracking. In ZebraNet, both sensor nodes and the BS are mobile. The sensor nodes are installed on the tracking collars of the zebras being tracked. These sensor nodes are custom-made wireless computing devices that are equipped with a GPS, a flash memory, a small CPU, a short-range radio, a long-range radio and a solar array of 14 cells. The BS, located on a moving vehicle (or a flying plane), is periodically driven-by (or flown-over) the network to collect the GPS locations of the sensor nodes. The communications between the sensor nodes in the network are carried out using short-range radio and the communications between the BS and the sensor nodes in the network are carried out using long-range radio. To allow data collection from those sensor nodes that cannot send data to BS using their own long-range radio, ZebraNet allows peer-to-peer data collections among sensor nodes, i.e. nodes that are far away from the BS can send their data to the BS via the nodes that are located between them. The focus of the work is on investigating how to maximise successful data collections by the BS from the mobile sensor nodes. For this, the authors have designed and evaluated two data collection protocols, one is flooding-based and the other is history-based. With the flooding-based protocol, each sensor node sends its GPS location to all its neighbouring sensor nodes. The neighbouring sensor nodes do the same and the process continues until the location data reaches the BS. With the history-based protocol, each sensor node selects and sends its location data only to a set of neighbouring sensor nodes, which are selected based on the sensor node's past experience, by measuring the data delivery ratios while using these nodes to send data to the BS. The evaluation results show that the two protocols perform differently under different network conditions. When sensor nodes have ample storage and bandwidth capacities, the flooding-based protocol gives better data delivery ratios than the history-based protocol. However, when sensor nodes

have limited storage capacity but ample bandwidth capacity, the history-based protocol outperforms the flooding-based protocol. When the sensor nodes have ample storage capacity but limited bandwidth capacity, the flooding-based protocol performs better for short-range communications and the history-based protocol performs better for long-range communications. From these results, the authors conclude that a protocol that takes an adaptive approach in response to different resource capacities and network conditions would produce optimal results in terms of increasing data delivery ratios and reducing energy consumption. Building on this work, Zhang *et al.* [7] reported their experiences in developing hardware and low-level software for ZebraNet.

Xu *et al.* [8], [9] proposed a WSN based animal monitoring application to detect animals' physical locations and to monitor their movements in a specific observation area, without mounting any wireless devices on the animals. The idea used is to divide the observation area into multiple virtual grids and place multiple sensor nodes in each grid. These sensor nodes in each grid form a cluster, headed by a cluster head. Each sensor node in a cluster records any appearances of the animals and forwards the recorded data to the cluster head in the cluster. The BS, mounted on an unmanned aerial vehicle (UAV), periodically flies over the observation area and collects data from each cluster head. By using the collected data, the BS estimates the physical locations of tracked animals in a particular grid. The main contribution of the paper is the design of a Markov decision based path planning method to ensure more data can be collected for a given period of time. It predicts the places in the observation area, where animal presence is more likely so that the UAV would visit these places more frequently. The path planning method was evaluated using two datasets: one with zebra movement traces and the other with leopard tortoise movement traces. The evaluation results showed that the Markov decision based path planning method offers a better performance than the other path planning methods, namely random, greedy and travelling salesman problem-based methods, in terms of reducing delays in data collections and increasing the number of animals that can be captured in the collected data.

B. EFFECTS OF WILDLIFE TRACKING DEVICES ON TRACKED ANIMALS

It has long been suggested that a tracking device placed on an animal can have a negative impact on the animal's behaviour, movement or even growth [21]–[23]. To reduce such impact, a set of guidelines have been established for the use of tracking devices on animals over the years. One important guideline is that the total weight of an attachment (includes collar, transmitter, battery, aerial and bonding material) should ideally be less than 5% of the animal's body mass (BM) [10]–[12]. However, two notable studies [13], [14] have found that, even within the acceptable weight limits set in the guidelines, the slightest difference in an attachment weight can have a significant impact on the movement of a tracked animal.

One of these notable studies was carried out by Brooks *et al.* [13]. The study investigated the effect of the weight of a tracking collar on a zebra's behaviour by measuring and comparing the behaviours of zebras wearing two different types of tracking collars. Both types of tracking collar were attached with a GPS and they differed slightly in their weights. Both types of tracking collars weighed less than 5% of a zebra's BM. The study results showed that the travelling speed of the zebras wearing the heavier collars was reduced by as high as 50% as compared to the zebras, wearing the lighter collars. It was also found that the difference in their speeds was more noticeable when the zebras were in their grazing state as compared to their fast-moving state. The authors emphasized that, to improve the accuracy of animal behaviour pattern studies, the smallest differences in their collar weight and fit should be taken into account when collecting and studying the data.

The second notable study was carried out by Couglin and van Heezik [14]. This study investigated the impact of a GPS collar weight on domestic cats' movements. Three types of collars with different weights were used in the investigation. The respective weights were: light – 30 g (<1% BM), medium – 80 g (∼2% BM), and heavy – 130 g (∼3% BM). The movements of 20 domestic cats, wearing the three different types of collars, were recorded over 3 weeks. Each week the collar weights were adjusted, so that each cat wore all the three types of collars in a random sequence. The recorded data were used to estimate the spatial extent of cats' movements (referred to as its home-range size) and distances travelled from home. The results showed that the home-range size and the distances travelled vary with the collar weights. When the cats wore the heavy collars, their home-range size is reduced and they travelled over smaller distances as compared to when they were wearing the other two types of collars. Based on this study, the authors recommended that the weight of tracking collars put on domestic cats should be no more than 2% BM so as not to restrict their usual movement patterns.

Based on the above related work analysis we can make the following observations: (i) there is a need to reduce the weights of tracking collars worn by animals as much as possible, (ii) existing solutions are largely GPS-based and use long distance communication facilities, which contribute to the weight of a tracking collar, and (iii) there is no systematic investigation as how to facilitate animal tracking by merely using a WSN consisting of ordinary sensor nodes without any support of GPS and long-range radio communications facilities. Although the work by Xu *et al.* [8], [9] is WSN based, the proposal only provides an animal monitoring function and can only detect animals' physical locations and monitor their movements within a cluster that is covered by short-range radios. To cover a larger geographical area and/or accommodate a higher level of sensor node density, a larger number of clusters and cluster heads need to be deployed. This not only increases deployment costs, but may also affects the scalability of the system. A larger number of clusters means that the time it takes for the BS to collect data from

all the cluster heads in the network will increase too. If this time is longer than the time it takes for the animals to roam from one cluster to another, then the monitoring results would not be accurate. The LMM system, to be presented in the remaining part of this paper, is designed to overcome these limitations.

III. DESIGN PRELIMINARIES

This section details the terminology, notations, assumptions and the requirements used in the description of the LMM system.

A. TERMINOLOGY AND NOTATIONS

Terminology and notations used to describe the LMM system are summarised in TABLE 1.

B. ASSUMPTIONS

The design has made use of the following assumptions:

(A1) The MDC node has ample energy and memory capabilities.

(A2) The SDC and DG nodes have limited energy and memory capabilities.

(A3) The entire network is partitioned into zones and each such zone is called a ZoneNet. The topology in a ZoneNet (i.e. the topological relationships among the DG nodes and between the DG nodes and the SDC node in the ZoneNet) are established prior to the data collection operations. These topological relationships are dynamic due to the mobility of the DG nodes. To capture these dynamic relationships, the Collection Tree Protocol (CTP) [24]–[26] is executed prior to each data collection operation. In the process of establishing the topological relationships, each node will also learn its hop distance from the SDC node(s) it is connected to.

(A4) The clocks of all the nodes (i.e. the DG, SDC and MDC nodes) are synchronised.

(A5) Energy costs are measured in milliAmpere (mA), giving the average current drawn by a sensor node over a time interval of 'T' seconds at a constant voltage (i.e. 3.3 Volts (V) [15], [27]). Typically energy consumption is measured in Joules (J) and is given as:

$$Energy = Voltage \times Current \times Time \quad (1)$$

As the voltage is constant, a sensor node's energy consumption is proportional to the current drawn over the time interval 'T', which is measured in mA [28].

(A6) A large number of DG nodes are used in the experiments so as to model wildlife tracking for zebras, which typically move in herds and super herds.[2]

(A7) Communication channels are reliable and error free. In other words, any message sent will be received by its intended recipient correctly. Data integrity and reliability issues are outside the scope of the design.

[2]A typical herd constitutes of a male and a group of 5-6 females with their foals, whereas a super herd is formed when multiple such herds come together. A super herd can be as large as 300 zebras [29].

TABLE 1. Terminology and notations.

Symbol	Meaning
DG	Data Generator, a sensor node placed on a zebra's tracking collar
SDC	Slave Data Collector (also referred to as a stationary BS), a sensor node mounted at a fixed location in the network
MDC	Master Data Collector (also referred to as a mobile BS), a node mounted in an unmanned aerial vehicle (UAV)
Backend Server	A machine, deployed at a remote site, to estimate movement traces from collected data
ZoneNet	A zone consisting of multiple DG nodes and one SDC node, organized in a tree structure headed by the SDC node; the network consists of multiple ZoneNets
n	No. of DG nodes in a ZoneNet
m	No. of SDC nodes in the network
o	Total No. of DG nodes in the network, where $o = m \times n$
R	Communication radius of a SDC/DG node
t_1	Time duration for data collection by MDC, i.e. within t_1 one round of data collections by MDC from all SDC nodes should be completed
t_2	Time duration for data collection by SDC, i.e. within t_2 one round of data collections by SDC from all DG nodes in the ZoneNet should be completed
ATL	Aggregation based Topology Learning
Q_{SDCi}	Query message sent by SDC_i
QR_{DG}	Query response message sent by DG
REQ_{MDC}	Request message sent by MDC
RES_{SDCi}	Response message sent by SDC_i
$PUSH_{MDC}$	Data push message sent by MDC
$Node\_ID_X$	ID of a node X
$BEF\text{-}ID_X$	IDs of a set of nodes called Branch-based Extended Family Nodes at X
$LNP\text{-}EF_X$	Leaf Nodes' Positions, i.e. the positions of leaf nodes in the Extended Family at X
TD_X	Topology Data packed at X, which contains hop-wise positions of X's child nodes
TI_{Hop}	Estimated time taken to collect data from nodes that are one hop away
h	Estimated number of hops in a ZoneNet
TI_{Zone}	Estimated time taken by a SDC node to collect data from all h hops in the ZoneNet, where $TI_{Zone} = h \times TI_{Hop}$
TR_{SDC}	Time reference value set by a SDC node
h_{DG}	Hop count of a DG node
TO_{DG}	Time out value set by a DG node on receiving Q_{SDCi}, where $TO_{DG} = TI_{Zone} - \{h_{DG} \times TI_{Hop}\}$
TI_{Zn}	Estimated time required by MDC node to collect data from one ZoneNet
LE	Location Estimation method
TR_{MDC}	Time reference value set by the MDC node
H[X][Y]	Hop distance between nodes X and Y
CM_X[i][j]	Connectivity matrix for X with i rows and j columns
Px[X]	x-coordinate of X's estimated physical location
Py[X]	y-coordinate of X's estimated physical location
Ox[X]	x-coordinate of X's real physical location
Oy[X]	y-coordinate of X's real physical location
DCC	Data Collection Coverage
VTD	Visit Time Duration
EE	Estimation Error

C. DESIGN REQUIREMENTS

This section specifies the requirements for the design of the LMM system. The requirements are of two groups: functional and performance requirements.

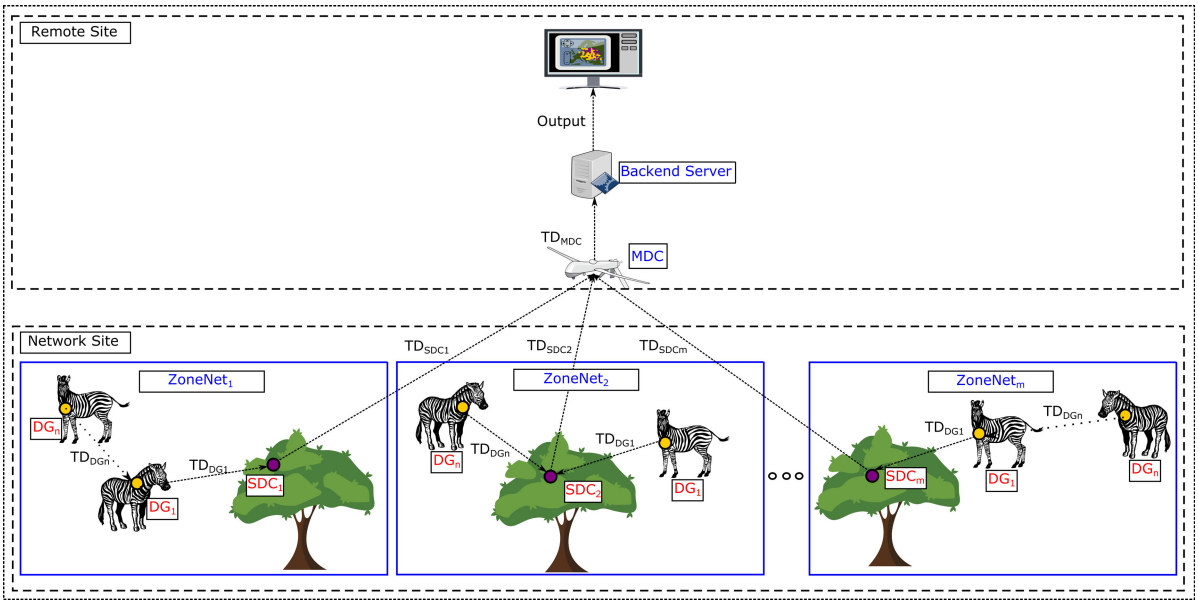


FIGURE 1. An overview of the LMM system architecture.

1) FUNCTIONAL REQUIREMENTS

(F1) The system should be able to collect topology data from as many sensor nodes as possible.

(F2) The system should be able to estimate the physical location of a sensor node based on the topology data collected at any moment of time. The sequence of a sensor node's physical locations estimated over time will plot the movement trace of the sensor node.

2) PERFORMANCE REQUIREMENTS

(P1) The computational cost imposed on sensor nodes should be as low as possible. To satisfy this requirement, functions carried out on sensor nodes should be minimal.

(P2) The storage requirements imposed on the sensor nodes should be as low as possible. To satisfy this requirement, aggregated or special coded data, not raw data, should be stored.

(P3) The communication overhead imposed on the sensor nodes should be as low as possible. To satisfy this requirement, the number and length of messages required to accomplish the data collection functions should be as few and as short as possible.

To satisfy the above performance requirements, a number of measures are used and these are: (i) each sensor node only performs essential functions, i.e. data collection, aggregation and transmission, (ii) data aggregation is used so as to make the length of data stored and transmitted independent of the volume of the data received and (iii) a special data structure is used to encode the topology data so as to maximise the number of sensor nodes from which the topology data can be collected for the given length of messages used to carry the data.

IV. THE LOCALIZATION AND MOBILITY MODELLING (LMM) SYSTEM: AN OVERVIEW

An overview of the LMM system architecture is shown in Figure 1. As shown in the figure, the system consists of a Backend Server, a mobile BS (called Master Data Collector (MDC) node), a set of stationary BSes (called Slave Data Collector (SDC) nodes), and sensor nodes (called Data Generator (DG) nodes). The network site (i.e. the entire observation area) is divided into multiple zones (called ZoneNets) each covered by a SDC node. Each SDC node collects topology data from all the 'n' DG nodes in its ZoneNet. This collection is done via hop-by-hop communication using the Aggregation based Topology Learning (ATL) protocol [30] that implements the idea of DCwEAS. The MDC node periodically travels to the Network Site, from the Remote Site, to collect the topology data from the 'm' SDC nodes on behalf of the Backend Server. This collection is carried out by using a peer-to-peer data collection protocol, called the MDC Data Collection protocol. The collected data are then transmitted to the Backend Server using another peer-to-peer transmission protocol, called the MDC Data Transmission protocol. Based on the received data, the Backend Server estimates the physical locations of the sensor nodes using a Location Estimation (LE) method. Periodical executions of these operations over time will allow the Backend Server to learn the movement traces of the DG nodes, thus tracking the movements of the animals.

Figure 2 summarises the LMM system operations. From the figure, we can see which protocol or method is used by which system component(s) and the sequence of operations from when the data are being collected to when movement traces are estimated based on the collected data. These operations can be classified into three phases (indicated in different colours). In the first phase, each SDC node in a ZoneNet

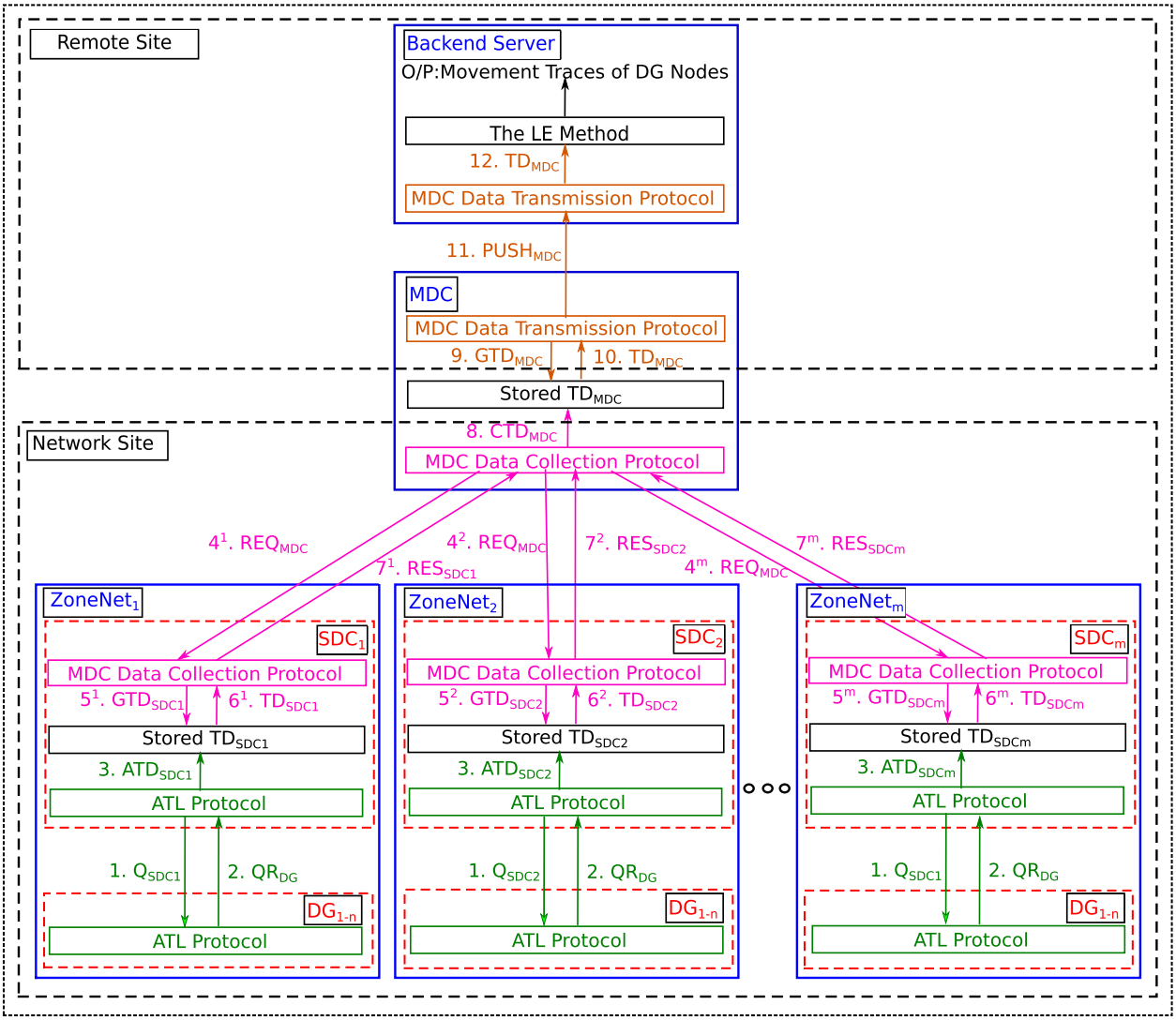


FIGURE 2. The LMM system architecture and an overview of its operations.

collects topology data from 'n' DG nodes in the ZoneNet using the ATL protocol. In this phase, a pull-based data collection approach is used, whereby the SDC node in a ZoneNet periodically initiates a topology data collection operation by broadcasting a query message (Q) in the ZoneNet and, in response to Q, each DG node in the ZoneNet sends its topology data, starting from the leaf nodes, hop-by-hop in a query response message (QR) towards the SDC node. In the second phase, the MDC node collects the topology data from the SDC nodes in the 'm' ZoneNets using the MDC Data Collection protocol. This phase too uses the pull-based data collection approach but in a round robin fashion, i.e. the MDC node pulls the topology data from each of the SDC nodes in a sequential order. In the third phase, the MDC node sends the collected topology data to the Backend Server using a MDC Data Transmission protocol. This phase uses a push-based data collection approach, i.e. the MDC node sends the topology data to the Backend Server without requiring any query message from the Backend Server. Once the topology data is received by the Backend Server, it estimates the movement traces for the DG nodes using the LE method. This Multi-Zone Multi-Hierarchy (MZMH) communication structure can, firstly, reduce the overall data collection time, as the data collections in different ZoneNets are carried out in parallel. Secondly, it can make the system more scalable, as the number of SDC nodes deployed can scale up and down based on the size of the observation area and/or the number of DG nodes in the observation area. Furthermore, as the DG nodes are mobile, they can move from one ZoneNet to another or even out of the network, or move from outside the network to a ZoneNet. However, due to periodic data collection in the MZMH communication structure, the changes in network topology arising from these movements could be captured in the collected topology data.

In the following section, we describe the LMM system in more details.

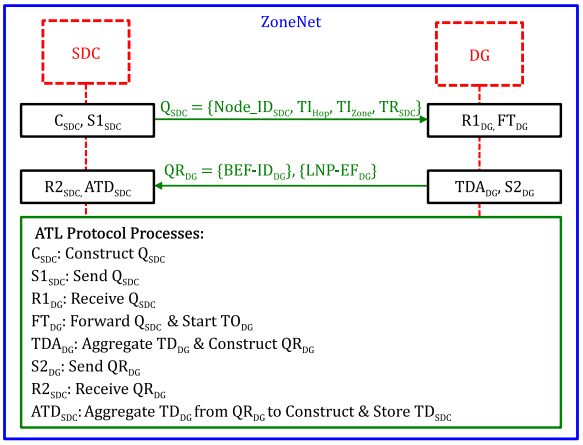


FIGURE 3. The ATL protocol.

V. THE LMM SYSTEM IN DETAIL

This section describes the three protocols, i.e. the ATL protocol, the MDC Data Collection protocol and the MDC Data Transmission protocol, which are used to facilitate the collection of topology data from sensor nodes by the Backend Server, and the LE method which is used to estimate the locations of the sensor nodes based on the collected data. The description covers the data structures, messages, functions and operations of the protocols and the LE method.

A. THE ATL PROTOCOL

The generic design of the ATL protocol has been described in [30]. Here, we describe how to apply the ATL protocol for topology data collections from the DG nodes in a ZoneNet. To start with, the SDC and the DG nodes in the ZoneNet are structured into a tree topology with the SDC node being the root of the tree. In this tree, the SDC node acts as the BS and each DG node sends its topology data, hop-by-hop, towards the SDC node, in a bottom-up fashion, i.e. starting from the leaf nodes of the tree. The topology data sent by a DG node captures the hop-wise position of the DG node from the SDC node in the tree. Let TD_X be the topology data collected at a node X (where X could be the SDC node or a DG node in the tree). TD_X captures the topology data for the entire tree (or sub-tree) headed by X. Assuming that X has b branches and each branch j (where $1 \leq j \leq b$) contains n_j nodes, then TD_X is expressed in the following data structure:

$$\begin{aligned} TD_X &= \{BEF - ID_X\}, \{LNP - EF_X\} \\ &= \{Node\_ID_X,\ Node\_IDs\ of\ n_1\ children\ on \\ &\quad branch\ 1,\ \ldots, Node\_IDs\ of\ n_b\ children\ on \\ &\quad branch\ b\},\ \{index\_number\ of\ leaf\ node\ n_1\ in \\ &\quad BEF - ID_X,\ \ldots,\ index\_number\ of\ leaf\ node \\ &\quad n_b\ in\ BEF - ID_X\} \end{aligned} \tag{2}$$

The components of the ATL protocol are summarised in Figure 3. As shown in the figure, the protocol consists of 2 messages, a request message Q_{SDC} and a response message QR_{DG}, along with 8 functions for the construction, sending, receiving and processing of the protocol messages. The SDC node periodically solicits topology data from the DG nodes by broadcasting the query message Q_{SDC} and receiving responses to the query message, i.e. QR_{DG}. The detailed operation of the ATL protocol is as follows.

- At the start of a time interval, say 't_2', the SDC node constructs and broadcasts a query message Q_{SDC} to its 1-hop downstream DG nodes, and in return, the SDC node expects to receive the topology data (i.e. TD_{DG}) of the entire ZoneNet, which will be forwarded to the SDC node via its 1-hop downstream DG nodes. A time reference and two time intervals are included in each query message. The time reference TR_{SDC} is a fresh estimate of the SDC node's clock for DG nodes to adjust their clocks with respect to the SDC node. One time interval is TI_{Hop}, which defines an estimated time for a DG node to collect data from one hop and the other time interval is TI_{Zone}, which defines an estimated time for the SDC node to collect data from the entire ZoneNet. In other words, by the expiry of TI_{Zone}, the SDC node should have received the solicited topology data from all the DG nodes in the ZoneNet.
- Upon the receipt of Q_{SDC}, each DG node forwards Q_{SDC} to its downstream neighbours and adjusts its clock based on TR_{SDC}. It then starts a timeout TO_{DG}, where the value of TO_{DG} is computed as a function of three values: the DG node's hop count, h_{DG}, TI_{Hop} and TI_{Zone}. The values of TI_{Hop} and TI_{Zone} are carried in Q_{SDC}. The process continues until Q_{SDC} reaches the leaf DG nodes of the tree. Each leaf DG node, upon the receipt of Q_{SDC}, sends a response QR_{DG}, which contains the topology data of the tree headed by the DG node (for the leaf node, the tree is null), to its 1-hop upstream DG node. The upstream DG node upon the receipt of the responses, which have been received by the expiry of TO_{DG}, aggregates the topology data contained in the responses along with its own topology data and packs the aggregated topology data into a new response message before sending it to its 1-hop upstream node. The aggregation is done by using equation (2). This process continues until the SDC node receives QR_{DG} from its 1-hop downstream DG nodes.
- Upon the expiration of the 't_2' interval, the SDC node aggregates the topology data contained in all the QR_{DG} messages, which are received during the interval, constructs TD_{SDC} containing the aggregated data and stores TD_{SDC} in its memory, waiting for data collection by the MDC node in the next stage. The aggregation here is done by using a local method ATD_{SDC} which implements the data aggregation function based on equation (2). The topology data contained in TD_{SDC} are the data of the entire tree headed by the SDC node.

B. THE MDC DATA COLLECTION PROTOCOL

The MDC Data Collection protocol is used by the MDC node to collect topology data from all SDC nodes in the network.

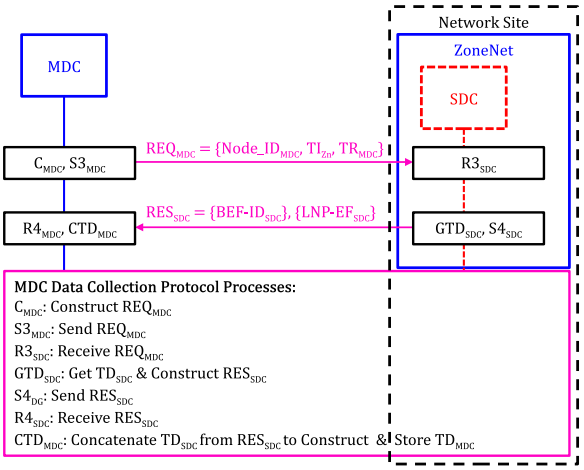


FIGURE 4. The MDC data collection protocol.

The MDC Data Collection protocol uses two data structures, TD_{SDC} and TD_{MDC}. TD_{SDC} is defined in equation (2), where X now is a SDC node. TD_{MDC} is a concatenation of the topology data received from all the SDC nodes in the network. Given 'm' SDC nodes, TD_{MDC} is defined as:

$$TD_{MDC} = \{BEF - ID_{SDC1}\}, \{LNP - EF_{SDC1}\}|| \{BEF - ID_{SDC2}\}, \{LNP - EF_{SDC2}\}|| \ldots || \{BEF - ID_{SDCm}\}, \{LNP - EF_{SDCm}\} \quad (3)$$

An overview of the MDC Data Collection protocol is given in Figure 4. As shown in the figure, the protocol consists of 2 messages, a request message REQ_{MDC} and a response message RES_{SDC}, along with 7 functions for the construction, sending, receiving and processing of the protocol messages. In detail, the operation of the protocol is as follows.

- At the start of a time interval, say 't1', the MDC node travels to the Network Site and pulls the topology data from the SDC nodes in each ZoneNet. The SDC nodes are pulled, one by one, in a round robin fashion. For each pulling, the MDC node constructs and transmits a request message REQ_{MDC} to a SDC node and receives a response message RES_{SDC} from the pulled SDC node. The response message contains the requested topology data TD_{SDC}. A time reference and a time interval are included in each request message. The time reference TR_{MDC} is a fresh estimate of the MDC node's clock for SDC nodes to adjust their clocks with respect to the MDC node. The time interval TI_{Zn} defines an estimated time for the MDC node to receive a response upon the transmission of a request message. In other words, by the expiry of TI_{Zn}, the MDC node should have received the response message from the pulled SDC node. If a response message is not received by the expiry of TI_{Zn}, the MDC node will resend the request message.
- When the SDC node receives REQ_{MDC}, it constructs a response message RES_{SDC}, which contains TD_{SDC}, i.e. the topology data of the tree headed by the SDC node, and sends the message to the MDC node. It also adjusts its clock based on TR_{MDC}.
- Once the MDC node has received the topology data from all the SDC nodes (i.e. from all the ZoneNets), it concatenates the data to produce TD_{MDC} using the local method CTD_{MDC}, which implements the data aggregation function based on equation 3, and stores TD_{MDC} in its memory ready for delivery to the Backend Server in the next stage. Upon the expiration of 't1', the MDC node returns back to the Remote Site and transmits TD_{MDC} to the Backend Server.

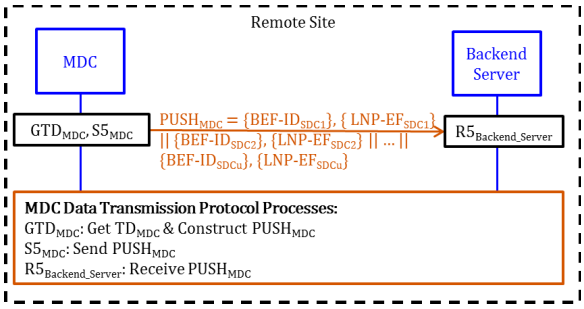


FIGURE 5. The MDC data transmission protocol.

C. THE MDC DATA TRANSMISSION PROTOCOL

The MDC Data Transmission protocol is used by the MDC node to transmit the collected topology data to the Backend Server. The protocol uses the data structure TD_{MDC}, defined in equation (3).

An overview of the protocol is given in Figure 5. As shown in the figure, the protocol consists of only one protocol message, i.e. $PUSH_{MDC}$, and 3 functions for the construction, sending and receiving of the protocol message. The operation of the protocol is as follows:

- To send collected topology data to the Backend Server, the MDC node performs two operations: (i) gets TD_{MDC} from its memory and constructs a $PUSH_{MDC}$ message containing TD_{MDC}, and (ii) sends $PUSH_{MDC}$ to the Backend Server.
- Upon the receipt of the $PUSH_{MDC}$ message, the Backend Server extracts TD_{MDC} from the message and uses it as an input of the LE method to estimate the physical locations of all the DG nodes captured in the received topology data.

D. THE LE METHOD

The LE method applies trilateration on the collected topology data to estimate the movement traces of the DG nodes on the Network Site. The LE method makes use of five data structures. The first data structure is TD_{MDC}, which is defined in equation (3) and captures the topology data of the DG nodes on the entire Network Site.

The second data structure H[][] captures the hop distances of a given node (SDC/DG) with respect to each of the other nodes in the network. With a total of 'm' SDC nodes and 'o'

TABLE 2. Connectivity Matrix for DG Node X.

Nodes	**SDC_1**	**SDC_2**	...	**SDC_k**	**DG_1**	**DG_2**	...	**DG_l**	**X**
Physical Locations	$\{x_1,y_1\}$	$\{x_2,y_2\}$	...	$\{x_k,y_k\}$	$\{x_{1'},y_{1'}\}$	$\{x_{2'},y_{2'}\}$	...	$\{x_l,y_l\}$	$\{x_u,y_u\}$
SDC_1	0	∞	...	∞	h1	∞	...	∞	h2
SDC_2	∞	0	...	∞	∞	h3	...	∞	h4
⋮									
SDC_k	∞	∞	...	0	∞	∞	...	h5	h6
DG_1	h1	∞	...	∞	0	∞	...	∞	h2-h1
DG_2	∞	h3	...	∞	∞	0	...	∞	h4-h3
⋮									
DG_l	∞	∞	...	h5	∞	∞	...	0	h6-h5
X	h2	h4	...	h6	h2-h1	h4-h3	...	h6-h5	0

*h1=H[SDC_1][DG_1], h2=H[SDC_1][X], h3=H[SDC_2][DG_2], h4=H[SDC_2][X], h5=H[SDC_k][DG_k], h6=H[SDC_k][X]

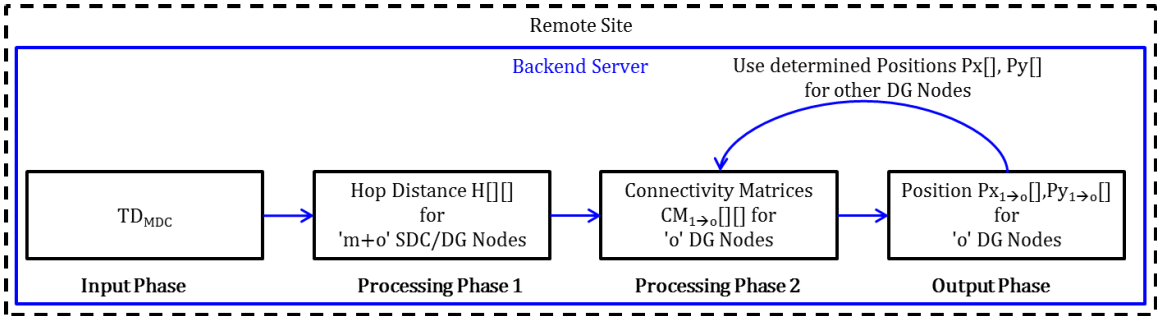


FIGURE 6. The LE method overview.

DG nodes (where o = m × n) on the Network Site, the size of H[][] is (m+o)×(m+o). For a DG node X connected with another SDC/DG node Y, H[][] is defined as:

$$H[Node\_ID_X][Node\_ID_Y] = Hop\ distance\ between\ X\ and\ Y \quad (4)$$

where the Node_IDs for the two connected nodes X and Y are used as indexes in H[][].

The third data structure CM_X[][] is the connectivity matrix for a DG node X. It specifies the hop distances of X with the SDC/DG nodes that are connected to X and the physical locations of these SDC/DG nodes. Table 2 shows the generic format of CM_X[][], where X is connected to 'k' SDC nodes through 'l' DG nodes. $\{x_i,y_i\}$ represents the known/estimated physical location of a SDC/DG node 'i', $\{x_u,y_u\}$ represents the unknown physical location of X, which is to be estimated by using the LE method, and h1, h2, h3, and so on refer to the hop distance between any two nodes in the connectivity matrix. CM_X[][] is an adjacency matrix for X and the 'k+l' SDC/DG nodes, with 0s at the diagonals, hop distances retrieved from H[][] for the connected nodes and an ∞ value for the remaining (i.e. unconnected) nodes. With a total of 'k' SDC nodes and 'l+1' DG nodes, CM_X[][] contains 'k+l+3' rows (one additional row for the physical locations of the nodes) and 'k+l+2' columns.

The fourth and fifth data structures, i.e. Px[X] and Py[X], respectively specify the estimated x-coordinate and y-coordinate, i.e. the physical location, of a DG node X on a 2D plane.

An overview of the operations of the LE method is summarised in Figure 6. As shown in the figure, the operations are classified into four phases, the Input Phase, Processing Phase 1, Processing Phase 2 and the Output Phase. The Input Phase is for the LE method to read the topology data received from the MDC node (i.e. TD_{MDC}). Processing Phase 1 is for calculating the hop distances H[][] between all SDC/DG nodes in the network and this calculation is based on the topology data contained in TD_{MDC}. Processing Phase 2 is for computing the connectivity matrix for each DG node X (i.e. CM_X[][]) using the hop distances H[][]. The Output Phase is for estimating the physical location of each DG node X (i.e. Px[X] and Py[X]) using CM_X[][].

It is worth-noting that the computations carried out in Processing Phase 2 and the Output Phase are iterative, as the output from a previous execution of the LE method (i.e. the physical location estimated for a DG node) may be used in computing connectivity matrices of some other DG nodes. These iterative computations continue until the physical locations of all the 'o' DG nodes have been estimated.

In the following, we describe the four phases of the LE method in detail. The LE method contains three algorithms,

which are the Hop Distance Calculation (HDC) Algorithm, the Connectivity Matrix Generation (CMG) Algorithm and the Physical Location Generation (PLG) Algorithm. In the following descriptions, we show how these algorithms are implemented and how these are used along with the data structures mentioned above to accomplish the task of estimating DG nodes' movement traces.

Input Phase: Getting Topology Data

In this phase, the LE method reads its input data, i.e. TD_{MDC}. This data is sent by the MDC node via the execution of the MDC Data Transmission protocol.

Processing Phase 1: Calculating Hop Distances

In this phase, the hop distances of each pair of the 'm+o' SDC/DG nodes are calculated. The algorithm used for this calculation is the Hop Distance Calculation (HDC) Algorithm. The pseudo code for the HDC Algorithm is given in Algorithm 1.

As shown in the pseudo code, the algorithm first separates the concatenated topology data in TD_{MDC} to obtain the topology data for each of the SDC node 'S'. Next, based on the obtained topology data, it determines the number of branches in BEF-ID_S and separates these branches using the index_numbers in LNP-EF_S. For each of the DG nodes contained in each of the branches, the algorithm calculates the hop distances between the SDC node and each DG node in the branch. It then stores the calculated hop distances in H[][], where Node_IDs of the two connected nodes (DG-DG, or DG-SDC) are used as indexes in H[][]. The process continues until H[][] is calculated for all the nodes present in TD_{MDC}.

Processing Phase 2: Generating Connectivity Matrices

In this phase, the connectivity matrices for each of the DG node X are computed using H[][] along with the known/estimated physical locations {Ox[], Oy[]} for the SDC/DG nodes that are connected with X and the Connectivity Matrix Generation (CMG) Algorithm. The pseudo code for the CMG Algorithm for X is given in Algorithm 2.

The working of the algorithm can be summarised into 2 steps. In step 1, the algorithm gets the values of the first two rows and the first column in CM_X[][]. To do so, it reads through the row H[X][1→'m+o'] and determines the 'Y' SDC/DG nodes that are connected with X. These SDC/DG nodes are referred to as the 'anchor nodes' of X. For each of the 'Y' anchor nodes, the algorithm determines its Node_ID and places it in the first row and first column in CM_X[][]. It also puts the known/estimated physical location {Ox[Y], Oy[Y]} for each of the 'Y' anchor node in the second row in CM_X[][], under its respective Node_ID.

In step 2, the algorithm calculates the hop distances for CM_X[][]. To do so, it iterates through the first row in CM_X[][] and gets two Node_IDs 'A' and 'B' in each iteration. It then uses the two Node_IDs as indexes to obtain the hop distance in H[A][B] data structure. If a value can be computed for H[A][B], the algorithm puts this value at the intersection of A and B in CM_X[][]. Otherwise, it puts an ∞ value at the intersection.

Algorithm 1 Hop Distance Calculation (HDC) Algorithm for 'm+o' SDC/DG Nodes

Input: $TD_{MDC} = \{(\text{BEF-ID}_{SDC1}, \text{LNP-EF}_{SDC1}) \,||\, (\text{BEF-ID}_{SDC2}, \text{LNP-EF}_{SDC2}) \,||\, \ldots ||\, (\text{BEF-ID}_{SDCm}, \text{LNP-EF}_{SDCm})\}$
Output: H[][]

For each SDC node 'S' (1≤S≤m):

Get Node_ID for S as:
    Y = BEF-ID_S[0]
Identify the number of branches b in BEF-ID_S.
    b = Total index_numbers in LNP-EF_S
i ← 1
For each b:
    hop_dist ← 1
    b_start_index ← i ▷ *start index of branch b*
    while i ≤ LNP-EF_S[b] **do**
        X = BEF-ID_S[i] ▷ *Node_ID of DG node at index i*
        H[X][X] = 0 ▷ *X's hop distance with itself*
        H[X][Y] = hop_dist ▷ *X's hop distance with SDC node Y*
        j = hop_dist - 1
        k = b_start_index
        while j ≥ 1 **do**
            Z = BEF-ID_S[k] ▷ *Node_ID of DG node at index k*
            H[X][Z] = j ▷ *X's hop distance with DG node Z*
            j = j − 1
            k = k + 1
        end while
        i = i + 1
        hop_dist = hop_dist + 1
    end while

Output Phase: Determining Physical Locations

In this phase, the physical location for each DG node X is estimated using CM_X[][] and the Physical Location Generation (PLG) Algorithm. The pseudo code for the PLG Algorithm for X is given in Algorithm 3. The algorithm is a variation of the DV-hop algorithm [17], with the following essential differences:

- In DV-hop algorithm, the anchor nodes are present at fixed locations in the network. However, the PLG algorithm uses a dynamic set of anchor nodes at any given time, comprising of both SDC nodes that are at fixed locations in the network and the DG nodes that are mobile in the network. This dynamic set of anchor nodes is determined from the collected topology data and is

Algorithm 2 Connectivity Matrix Generation (CMG) Algorithm for a DG Node X

Input: H[][], Ox[], Oy[]
Output: CM_X[][]

1: Get Node_IDs, Ox[Y] and Oy[Y] for 'Y' nodes connected with X and generate first two rows and first column in CM_X[][].
2: total_nodes ← m + o
3: count ← 0
4: **for** i = 0, i < total_nodes, i++ **do**
5: **if** H[X][i] ≠ { } **then** ▷ *Is X connected to i?*
6: CM_X[0][count+1] = i ▷ *Put Node_ID 'i' in first row*
7: CM_X[1][count+1] = Ox[i] || Oy[i] ▷ *Put physical location of 'i' in second row*
8: CM_X[count+2][0] = i ▷ *Put Node_ID 'i' in first column*
9: count = count + 1
10: **end if**
11: **end for**
12: Generate the remaining rows in CM_X[][] using H[][] and first row in CM_X[][].
13: **for** i = 0, i < count, i++ **do**
14: A = CM_X[0][i+1] ▷ *Get Node_ID from first row*
15: **for** j = 0, j < count, j++ **do**
16: B = CM_X[0][j+1] ▷ *Get Node_ID from first row*
17: **if** H[A][B] ≠ { } **then** ▷ *Is A connected to B?*
18: CM[i+2][j+1] = H[A][B] ▷ *Put hop distance between A and B in CM[][]*
19: **else**
20: CM[i+2][j+1] = ∞
21: **end if**
22: **end for**
23: **end for**

present as the first row and first column in CM_X[][] for any DG node X.

- The DV-hop algorithm uses the shortest path algorithm to determine the hop distances between anchor nodes and the node whose location is to be determined. The PLG algorithm uses the hop distances between anchor nodes (both SDC and DG) and the DG node X that are determined dynamically by applying HDC and CMG algorithms on the collected topology data.

The operation of the PLG algorithm can be summarised into 8 steps. In the first 3 steps, the algorithm reads the physical locations of the anchor nodes and the hop distances contained in CM_X[][].

In the next two steps, it updates the hop distances in CM_X[][]. In detail, mutual distances between any two different nodes and their communication radii are used to update the ∞ hop distances in any cell of CM_X[][] (step 4) and then the shortest path algorithm is applied to calculate a minimum hop distance between any two connected nodes (step 5).

Algorithm 3 Physical Location Generation (PLG) Algorithm for a DG Node X

Input: CM_X[][]
Output: Px[X], Py[X]

1: c ← No. of columns in CM_X[][]
2: Get physical locations of anchor nodes from CM_X[][]:
3: **for** i = 1, i < c − 1, i++ **do** ▷ *starting from second column and going till second last column*
4: 2.1.1. x[i−1] ← x-coord in CM[1][i]
5: 2.1.2. y[i−1] ← y-coord in CM[1][i]
6: **end for**
7: Get hop distances from CM_X[][]:
8: **for** i = 2, i ≤ c, i++ **do** ▷ *starting from row 2*
9: **for** j = 1, j < c, j++ **do** ▷ *starting from column 1*
10: dhop[i−2][j−1] ← CM[i][j]
11: **end for**
12: **end for**
13: Update *dhop* values for non-connected nodes (i.e. those with an ∞ value):
14: **for** i = 0, i < c−1, i++ **do**
15: **for** j = 0, j < c−1, j++ **do** ▷ *computing mutual distance between i and j and comparing with R*
16: **if** (dhop[i][j] == ∞ AND $\sqrt{(x[i]-x[j])^2+(y[i]-y[j])^2} \le$ R) **then**
17: 4.1.1.1. dhop[i][j] = 1
18: **end if**
19: **end for**
20: **end for**
21: Apply shortest path algorithm to compute minimum *dhop* value at each connection:
22: **for** k = 0, k < c-1, k++ **do**
23: **for** i = 0, i < c-1, i++ **do**
24: **for** j = 0, j < c-1, j++ **do**
25: **if** dhop[i][k] + dhop[k][j] < dhop[i][j] **then**
26: 5.1.1.1. dhop[i][j] = dhop[i][k] + dhop[k][j] ▷ *getting the shortest hop distance between i and j*
27: **end if**
28: **end for**
29: **end for**
30: **end for**
31: Compute *HopSize*[0→c-2] values for anchor nodes.
32: **for** i = 0, i < c − 2, i++ **do**
33: $\text{HopSize}[i] = \frac{\sum_{j=1}^{c-2} \sqrt{(x[i]-x[j])^2+(y[i]-y[j])^2}}{\sum_{j=1}^{c-2} dhop[i][j]}, i \neq j$
34: **end for**
35: Calculate X's distance d[0→c−2] from each anchor node.

Algorithm 3 *(Continued.)* Physical Location Generation (PLG) Algorithm for a DG Node X

36: **for** i = 0, i < c − 2, i+Z **do**
37: d[i] = HopSize[i] × dhop[i][c−1]
38: **end for**
39: Compute Px[X], Py[X] with the least squares solution using x[0→c−2], y[0→c−2] and d[0→c−2].

$$\begin{bmatrix} Px[X] \\ Py[Y] \end{bmatrix} = (\mathrm{A}^T\ \mathrm{A})^{-1}\mathrm{A}^T\ \mathrm{B}, \text{ where:}$$

$$A = 2 \times \begin{bmatrix} x_0 - x_{c-2} & y_0 - y_{c-2} \\ x_1 - x_{c-2} & y_1 - y_{c-2} \\ \vdots & \vdots \\ x_{c-3} - x_{c-2} & y_{c-3} - y_{c-2} \end{bmatrix}$$

$$B = \begin{bmatrix} x_0^2 - x_{c-2}^2 + y_0^2 - y_{c-2}^2 + d_{c-2}^2 - d_0^2 \\ x_1^2 - x_{c-2}^2 + y_1^2 - y_{c-2}^2 + d_{c-2}^2 - d_1^2 \\ \vdots \\ x_{c-3}^2 - x_{c-2}^2 + y_{c-3}^2 - y_{c-2}^2 + d_{c-2}^2 - d_{c-3}^2 \end{bmatrix}$$

In the final 3 steps, the DV-hop algorithm is applied to estimate X's physical location. In detail, for each anchor node, an average physical distance (in meters) for one hop is estimated (step 6), the physical distances (in meters) between X and each of the anchor nodes are estimated (step 7) and physical location of X is estimated using the least squares solution (step 8).

VI. SIMULATION STUDY

This section presents simulation studies of the LMM system. The studies are in two parts. In part one (i.e. Study-1), we investigate the effects of different system parameter value settings on the performance of the system, where the performance is measured in terms of the number of DG nodes from which topology data can be collected by the LMM system. The purpose of this study is two-fold. The first is to find an efficient system architecture that could maximize the amount of topology data that could be collected by the system. This covers how to collect topology data, how many data collectors should be used and what mobility models should be used on these data collectors. The second is to investigate the optimum parameter value settings with which the system could collect topology data from a maximum number of DG nodes. The parameters that have been investigated are communication radius between the data collector and the DG nodes in a ZoneNet, the payload lengths of the protocol messages, the number of SDC nodes that should be deployed and how they should be positioned on the Network Site. In the second part (i.e. Study-2), we will adopt the system architecture and the optimum parameter value settings, discovered in Study-1, to the LMM system and analyse the costs and study the performance of the system in estimating the movement traces of DG nodes on the Network Site.

A. STUDY-1: DEFINING SYSTEM ARCHITECTURE AND THE PARAMETER VALUE SETTINGS

In the following, we first describe the environment under which the simulation was carried out and how simulation experiments were carried out and discuss the results and findings from these experiments.

1) SIMULATION ENVIRONMENT

The simulation is done using the Cooja simulator [31]. The simulated network consists of 100 DG nodes and varying numbers of SDC nodes (the number of the SDC nodes was set to 1, 4, 8, 16, respectively, in different experiments). The routing protocol used is the CTP (Collection Tree Protocol) [24]–[26]. All the SDC and DG nodes are programmed with the ATL protocol and the implementation of the ATL protocol is done on TinyOS [16] using nesC [32]. The implementation of the MDC Data Collection and the MDC Data Transmission protocols is done using VBA scripting in MS Excel.

The DG nodes are mobile and their movement traces are generated using the CRAWDAD ZebraNet dataset [18]. In total, 20 experiments have been carried out. In these experiments, we have studied the effects of using varying numbers of SDC nodes, different mobility models, both stationary and mobile and in the case where the SDC nodes are set to mobile, both random and non-random mobility models are investigated. The message payload length for Q_{SDC} is fixed to 14 bytes and the message payload length for REQ_{MDC} is fixed to 10 bytes. However, two message payload lengths, i.e. 27 bytes and 42 bytes, are considered for QR_{DG} and RES_{SDC}. In addition, three radio ranges, i.e. 50 m, 100 m and 200 m, are investigated. The time duration for each experiment run was set to 1120 seconds. This value is chosen via experiments and it is to ensure that the topology data is collected after the system is in a stable state. The 1120 seconds are further divided into 224 intervals of 5 seconds each and one topology data query is issued in each interval. In other words, each node can generate a maximum of 224 response messages during one experiment run.

The topology data collected at each SDC node was read into MS Excel, which was processed to generate TD_{MDC}.

The parameter value settings used in this study are summarised in Table 3.

2) RESULTS AND DISCUSSIONS

In this study, a total of 20 experiments were carried out with different parameter value settings to measure the Data Collection Coverage (DCC) percentage, which is defined as the percentage of DG nodes captured in the collected topology data to the total number of DG nodes in the network. In mathematical terms, DCC can be expressed as:

$$DCC = \frac{Number\ of\ DG\ Nodes\ in\ TD_{MDC}}{o} \times 100 \qquad (5)$$

where o represents the total number of DG nodes in the network. The 20 experiments were grouped into four sets. Each set led to an improved design for the next set of experiments.

TABLE 3. Simulation settings.

Parameter	Value
Protocols	ATL Protocol, MDC Data Collection Protocol, MDC Data Transmission Protocol
Development Platform	TinyOS-2.1.2 & nesC, MS Excel VBA Scripting
Simulator	Cooja
Number of Nodes	100 DG nodes, {1, 4, 8, 16} SDC nodes, 1 MDC node
Field Size	10,000 m^2
Node Distribution	100 x 100 grid
DG Placement & Movement	Mobile, Random Movement (CRAWDAD ZebraNet Dataset)
SDC Placement & Movement	Fixed (Top centre), No Movement Mobile, {Random Movement, Planned Movement}
Q_{SDC} Payload Length	14 bytes
QR_{DG} Payload Length	{27, 42} bytes
REQ_{MDC} Payload Length	10 bytes
RES_{SDC} Payload Length	{27, 42} bytes
Radio Range	{50, 100, 200} m
Number of Experiments	20
Average Simulation Time	1120 s

The Multi-Zone Multi-Hierarchy (MZMH) communication structure for the LMM system was defined based on the results from the final set of experiments.

Experiment Set I: Using One Stationary SDC Node

Figure 7 shows the network set up for Experiment Set I. As shown in the figure, the network consists of just one stationary SDC node and 100 DG nodes. The location of the SDC node is fixed at the top centre of the network. Two different message payload lengths and three different radio ranges were investigated. The experimental results under these parameter value settings are summarized in Table 4.

From the results shown in the table, it can be seen that the DCC percentage is significantly affected by the radio range but slightly affected by the message payload length. The longer the radio range, the higher the DCC percentage. When the radio range increases from 50 m to 200 m, the DCC percentage increases from 0% to 23% with the message payload length of 27 bytes and from 3% to 27% with the message payload length of 42 bytes. This is because, when the radio range is shorter, more relays are required for the data to travel from the DG nodes to the SDC node. More relays indicate that more messages will need to be handled by the DG nodes and for a given period of time and message length (carrying the data structure shown in equation 2), data from fewer DG nodes can be delivered to the SDC node. Further investigation has revealed that, with a message payload length of 27 bytes, an ATL message can only carry topology data from a maximum of 12 DG nodes. When the message payload length is increased to 42 bytes, this number increases to 24 DG nodes. Once the message payload length is reached, any data from further downstream DG nodes was simply left out. This means that the topology data relayed by a DG node may not capture all the data it has collected.

From the above results, we hypothesis that, the more the number of DG nodes from which topology data can be directly obtained by the SDC node, the higher the DCC percentage that may be achieved. To test this hypothesis, we have carried out the second set of experiments, Experiment Set II.

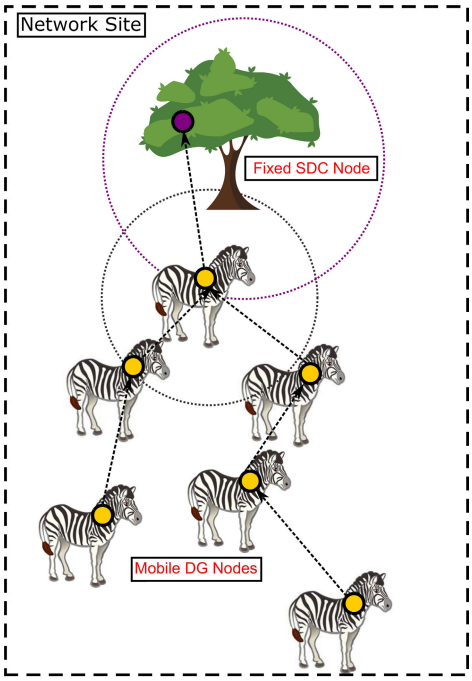


FIGURE 7. Network set up for experiment Set I.

Experiment Set II: Using One Mobile SDC Node With Random Movement

Figure 8 shows the network set up for Experiment Set II. As shown in the figure, the network also consists of one SDC node and 100 DG nodes. However, the SDC node is set to mobile and follows a random movement pattern, with a Visit Time Duration (VTD) value of 8 seconds, where a VTD value indicates the length of time in seconds for the SDC node staying in one ZoneNet. The random movement pattern is generated based on the CRAWDAD ZebraNet dataset. Similar to the Experiment Set I, two different message payload lengths and three different radio ranges were investigated. The results are summarized in Table 4.

The results show that, with the exception of case 1 (with the message payload length of 27 bytes and radio range of 50 m), the DCC percentages are lower than those from Experiment Set I. For example, with the radio range set to 200 m, the DCC percentage dropped from 23% to 11% (with the message payload length of 27 bytes) and from 27% to 15% (with the message payload length of 42 bytes). It is clear that the mobility of the SDC node causes this drop in DCC percentage. With CTP, the routing protocol used in the LMM system, a node forms routing connections with its neighbouring nodes and these connections are directed towards the BS (which is the SDC node in this experiment set up). Using these connections, the node sends its data towards the BS. Each time when the SDC node (i.e. the BS) moves, the node would need to adjust the connections. These adjustments cause two effects. Firstly, they take precious

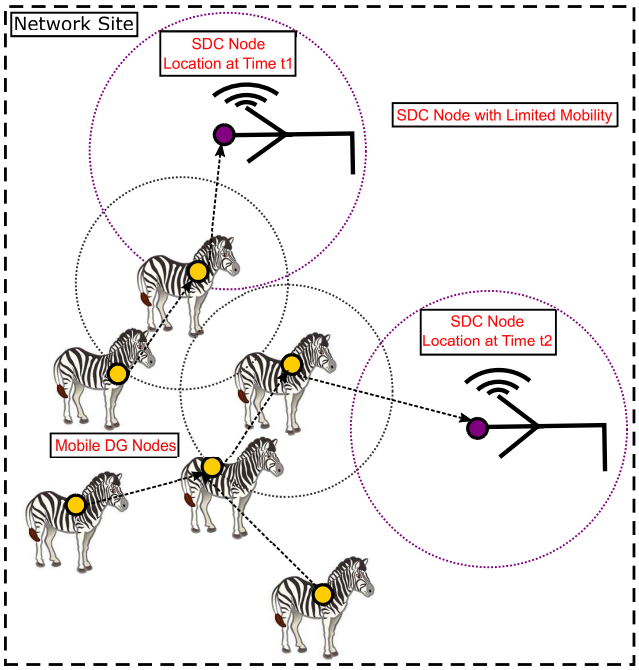


FIGURE 8. Network set up for experiment Set II.

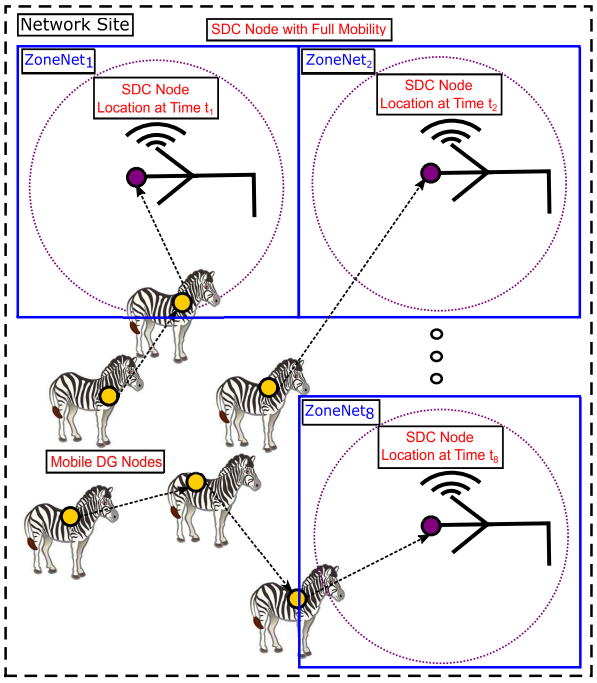


FIGURE 9. Network set up for experiment Set III.

communication times. As a result, the SDC node is not able to devote its time to obtain topology data directly from the DG nodes in the network. During the experiments, we also noted that the SDC node could only visit a subset of the nodes in the network before the collection time is over. Secondly, the connection adjustments may interrupt or terminate some already established data transmissions, resulting in the loss of topology data.

Lessons learnt from the two experiment sets let us believe that, while the mobility of the SDC node is necessary to reduce the number of data relays (or to increase direct communications between the SDC node and DG nodes), the negative effect of communication interruptions caused by the SDC node's mobility should be reduced as much as possible. To test this belief, we have carried out the third set of experiments, Experiment Set III, in which the whole network is divided into multiple zones, and the SDC node visits each of these zones in a round robin fashion, connecting with and collecting data from any DG nodes covered within its radio range.

Experiment Set III: Using One Mobile SDC Node With Planned Movement

Figure 9 shows the network set up for Experiment Set III. The network is divided into 8 zones ($\frac{1000\times1000}{\pi\times200^2} = 7.96$[3]), i.e. 8 ZoneNets. Similar to the experiments above, the network consists of 100 DG nodes. There is one SDC node that moves from one ZoneNet to the next in a round robin fashion. As shown in the figure, the SDC node started from the top left, it then moved left to right, from top to bottom. Five values of VTD (i.e. {4, 8, 16, 32, 64} seconds) were investigated. In Experiment Set III, the message payload length of 42 bytes was used and the radio range was set to 200 m. The use of these values was chosen as, in the previous two experiment sets, the maximum DCC percentage was achieved under these values. The results are summarized in Table 4.

From the results, we can see that VTD plays an important role in maximising the DCC percentage. Initially when the VTD value increases, the DCC percentage also increases. However, if the VTD value increases further, the DCC percentage decreases. Under the given experimental settings, the VTD value of 32 seconds produces the highest DCC percentage which is 53%, i.e. just over half of the DG nodes get their topology data delivered to the SDC node successfully. However, when the VTD value is further increased to 64 seconds, the DCC percentage decreases to 43%. This reduction can be explained as follows. Given the simulation time of 1120 seconds, 8 ZoneNets to visit and a VTD value of 64 seconds per ZoneNet, the SDC node can visit all the ZoneNets twice ($1120/(8 \times 64) = 2.18$). However, with the VTD value of 32 seconds, the SDC node can go around all the ZoneNets four times ($1120/(8\times32) = 4.38$). The more rounds the SDC node visits each ZoneNet, the more DG nodes it can get connected to, thus the higher the DCC percentage.

The findings from the above experiments have led to the use of a Multi-Zone Multi-Hierarchy (MZMH) communication structure in our LMM system. With this structure,

[3]Nitesh and Jana [33] compute a lower bound on the optimal number of anchor nodes required to completely cover a network, without any overlap. For a dense and uniform deployment of sensor nodes, with radio range R, in a network of dimensions $length \times breadth$, the optimal number of anchor nodes is computed as:

$$Optimal\_Number\_of\_Anchor\_Nodes = \frac{length \times breadth}{\pi \times R^2} \quad (6)$$

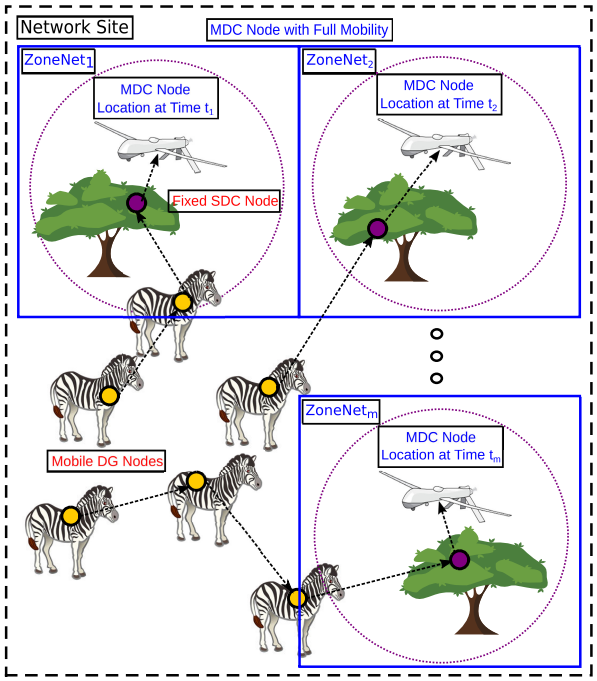


FIGURE 10. Network set up for experiment Set IV.

multiple BSes are used. The BSes are layered at two levels: at the top level is a master BS and at the bottom level are multiple slave BSes, one for each ZoneNet. The slave BSes (i.e. the SDC nodes) are stationary nodes, which are responsible for collecting topology data from DG nodes that are present in, or roam into, their respective ZoneNets. The mobile Master BS (i.e. the mobile MDC node) is responsible for collecting data from the SDC nodes. The mobile MDC node periodically travels to each ZoneNet to collect data from the SDC node. Experiment Set IV is designed to investigate the effectiveness of this MZMH approach.

Experiment Set IV: Using One Mobile MDC Node with Planned Movement and Multiple Stationary SDC Nodes, One per ZoneNet

Figure 10 shows the network set up for Experiment Set IV, which contains one MDC node, variable numbers of SDC nodes (i.e. 4, 8 and 16) and 100 DG nodes. We experiment with different numbers of ZoneNets in the network, i.e. {4, 8, 16}, where each ZoneNet contains a SDC node at a fixed location. The MDC node follows a planned movement pattern to cover the entire network, changing its location once every 32 seconds, and collects data from the SDC node in each ZoneNet. As in Experiment Set III, the message payload length of 42 bytes and the radio range of 200 m are used. The results are summarized in Table 4.

From the table, it can be seen that, with 4 SDC nodes, DCC percentage maximizes at 89%. This suggests that 4 SDC nodes are not sufficient to cover the entire network. With 8 and 16 SDC nodes in the network, we get a 100% DCC percentage in both cases. These results indicate that with the use of MZMH communication structure and provided that there are sufficient SDC nodes covering the whole network, we can collect data from all the sensor nodes in the network.

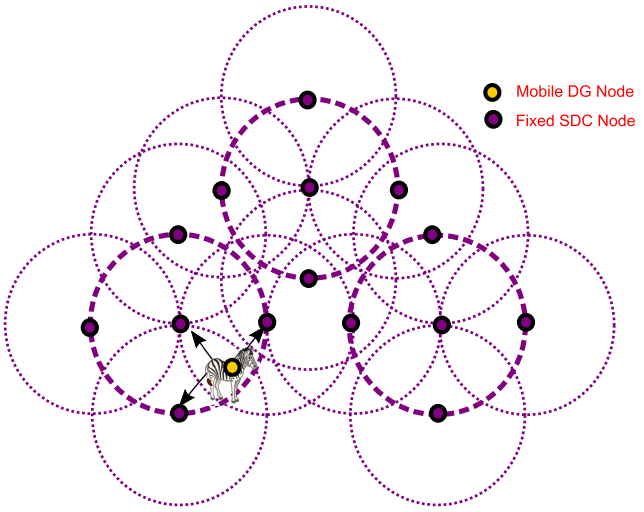


FIGURE 11. Proposed strategy for Study-2.

B. STUDY-2: ESTIMATING MOVEMENT TRACES

In this study, we evaluate the effectiveness of the LE method by applying it to estimate the movement traces of DG nodes in a network based on the topology data collected using the protocols described in Section V.

1) SIMULATION ENVIRONMENT

The initial network setup is identical to that used in Experiment Set IV, and the network consists of 100 DG nodes and 8 SDC nodes. The message payload length for REQ_{MDC} is 10 bytes, whereas the message payload length used for RES_{SDC} is 42 bytes, with the radio range fixed at 200 m. 30 unique TD_{MDC} datasets were collected in the simulation time of 1120 seconds. For each DG node X, we used the HDC and CMG algorithms (implemented using VBA scripting in MS Excel) and the TD_{MDC} datasets to get the hop distances H[][] and connectivity matrix CM_X[][].

During this process, we have discovered that the physical locations for some of the DG nodes cannot be estimated, as there are not sufficient anchor nodes (i.e. $\geq$3) for these nodes (their CM_X[][] matrices are not complete). This indicates that additional SDC nodes should be deployed so as to ensure each DG node could be captured by at least three SDC nodes, i.e. to allow range overlaps between different ZoneNets. For this reason, we have revised the network setup into the one shown in Figure 11 by adding four additional SDC nodes per each existing SDC node.

With the latest network setup and the updated CM_X matrix, we estimate the physical locations, i.e. Px[X] and Py[X], for each DG node X using the PLG algorithm (implemented as a modified version of a MATLAB script available for DV-hop algorithm [34]). Then, by plotting the Px[X] and

TABLE 4. Results for experiment Sets I-IV.

	No. of Nodes	Message Payload Length (bytes)	Radio Range (m)	Visit Time Duration (s)	Data Collection Coverage (%)
Experiment Set I	100	27	50	-	0
			100	-	5
			200	-	23
		42	50	-	3
			100	-	5
			200	-	27
Experiment Set II	100	27	50	8	3
			100	8	5
			200	8	11
		42	50	8	3
			100	8	5
			200	8	15
Experiment Set III	100	42	200	4	27
				8	32
				16	31
				32	53
				64	43
Experiment Set IV	104	42	200	32	89
	108	42	200	32	100
	116	42	200	32	100

TABLE 5. Simulation settings.

Parameter	Value
Algorithms	HDC Algorithm 1, CMG Algorithm 2, PLG Algorithm 3
Development Platform	MS Excel VBA Scripting, MATLAB
Simulator	MATLAB
Number of Nodes	100 DG nodes, 8 + 32 SDC nodes, 1 MDC node
REQ_{MDC} Payload Length	10 bytes
RES_{SDC} Payload Length	42 bytes
Radio Range	200 m
Number of Experiment Sets	2
Number of Estimated Locations	1757

Py[X] values estimated for X over time, we get X's movement trace. With a total of 30 TD_{MDC} datasets, we have estimated a total of 1757 physical locations, which correspond to the movement traces for 100 DG nodes.

The parameter values used in Study-2 are summarised in Table 5.

2) RESULTS AND DISCUSSIONS

Two sets of experiments (Experiment Sets V and VI) were carried out to evaluate the effectiveness of the LE method in estimating the movement traces of the DG nodes with the collected topology data. The purpose of carrying out these two experiments is to investigate if there is any benefits in using DG nodes as anchor nodes. In Experiment Set V, both SDC and DG nodes were used as anchor nodes (i.e. the physical locations of both SDC and DG nodes were used) when estimating the movement trace of a DG node X. In Experiment Set VI, only SDC nodes were used as anchor nodes when estimating X's movement trace. We have compared the results from both sets of experiments using a metric, called an Estimation Error (EE) value. The EE value measures the difference, in meters, between a set of physical locations in an estimated movement trace of a DG node and the same set of physical locations obtained from the node's real movement trace contained in the original dataset. Using the Root Mean Square Error (RMSE) formula, the EE value for X is given as:

$$EE_X = \frac{\sum_{i=1}^{d} \sqrt{(Px[i] - Ox[i])^2 + (Py[i] - Oy[i])^2}}{d} \quad (7)$$

where d represents the total number of physical locations in the X's estimated movement trace.

Experiment Set V: Using Both SDC and DG Nodes as Anchor Nodes

Figure 12 shows the simulation results of Experiment Set V. The figure plots both the physical locations of the DG nodes (i.e. their Px[] and Py[] values) in their estimated movement traces and their physical locations in their real movement traces. From the figure, we make the following observations:

- The estimated movement traces carry a total of 1757 physical locations in the 30 TD_{MDC} datasets, as compared to a total of 4704 physical locations in their real movement traces.
- The estimated movement traces follow the same patterns as the nodes' real movement traces.

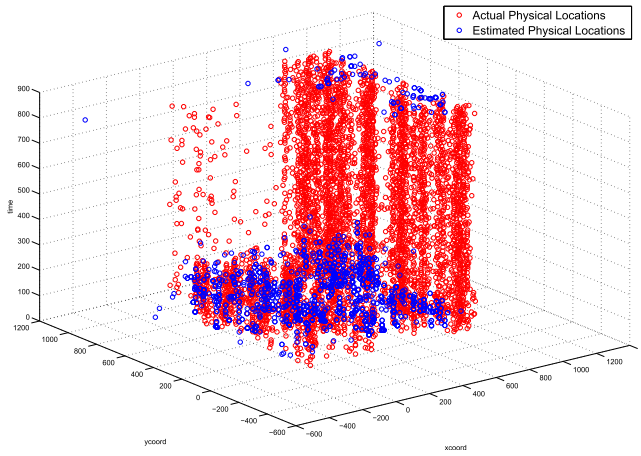


FIGURE 12. Experiment Set V: Using both SDC and DG nodes as Anchor nodes.

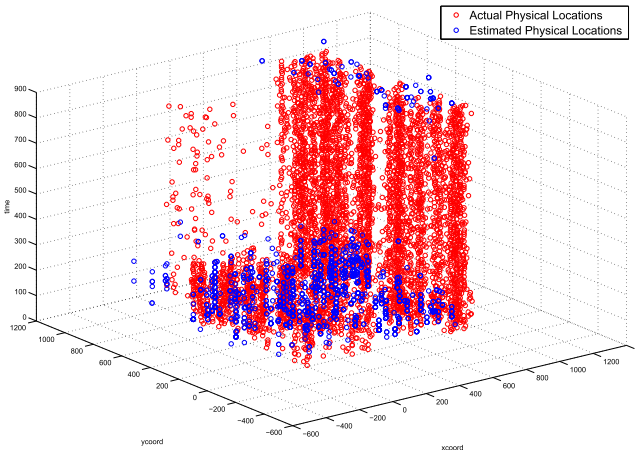


FIGURE 13. Experiment Set VI: Using only SDC nodes as anchor nodes.

- The estimated movement traces of different DG nodes carry different numbers of physical locations (i.e. 3 - 39). Each such number indicates the number of times the DG node's topology data appears in TD_{MDC} datasets.
- The estimated movement traces do not carry any physical locations in the time interval of 200 - 844 seconds during a simulation run. As TD_{MDC} datasets only contains the topology data collected by the SDC nodes, this means that the topology data maintained at the SDC nodes were not updated on the Cooja simulator during this time interval.

Experiment Set VI: Using Only SDC Nodes as Anchor Nodes

Figure 13 shows the simulation results obtained from Experiment Set VI. By comparing these results with those shown in Figure 12, we can see that, while the overall patterns of the two sets of results seems identical, results in Figure 13 are less spread out as compared to those in Figure 12. This is because, in Experiment Set VI, many physical locations had identical values, i.e. they were estimated to have the same values multiple number of times.

We now determine the EE values for the estimated movement traces of the DG nodes in Experiment Sets V and VI. Using equation (7), we have found that the average EE values for both the experiment sets are almost the same, and they are: 147.42 m (ranging from 9.69 m - 723.97 m) for Experiment Set V and 148.84 m (ranging from 17.70 m - 636.14 m) for Experiment Set VI. Overall, out of 100 DG nodes, the EE values of 63 DG nodes are lower in Experiment Set V as compared to Experiment Set VI. This means that, there are some cases, where using DG nodes as anchor nodes does not help to improve the accuracy in the estimations. To identify these cases, we have investigated the relationship between the number of anchor nodes and the EE values.

Figure 14 compares the EE values obtained in Experiment Sets V and VI against the number of anchor nodes, the result for the SDC anchor nodes and for the DG anchor nodes are shown separately. From the figure, we can observe that when the number of SDC nodes is 3, we get a higher EE value for a DG node from Experiment Set V as compared to Experiment Set VI. We have identified 10 such cases in the figure and this observation is found true for 8 out of the 10 cases. This means that, the movement traces of a DG node can be estimated more accurately with ≥ 3 SDC anchor nodes at fixed locations in the network, as compared to when the movement traces are estimated using mobile DG nodes as anchor nodes. Using only SDC anchor nodes for these 10 DG nodes, we are able to reduce the average EE value for Experiment Set V to 141.53 m. We were unable to determine any such relationship between the EE value for a DG node and the number of DG based anchor nodes used by the DG node.

During the investigation, we have also identified an interesting case where a DG node 'X' did not move but its parent node (a DG node 'Y') carrying its topology data, moved. As 'Y' gets connected to a new SDC node 'Z', either directly or via other DG nodes, it transferred its collected topology data to 'Z'. Based on the received data, 'Z' learned that 'X' was connected to it via 'Y', which is not true. In such a case, the hop distance values determined using the topology data received by Z would not be correct, thus the movement traces estimated, using these values, would be inaccurate as well.

To investigate this case further, we checked the CM_X matrices for the DG nodes with high EE values to identify any sudden changes in their hop distance values. More specifically, we checked the CM_X matrices for 20 DG nodes {9, 18, 22, 29, 38, 42, 50, 51, 52, 55, 58, 60, 63, 71, 79, 80, 81, 84, 91, 94}, which have EE values ranging from 194.17 m to 702.65 m. In the CM_X matrices for these DG nodes, we eliminated the rows and columns with the sudden hop distance changes and re-estimated the movement traces for these DG nodes. With the new estimations, we were able to reduce the EE values

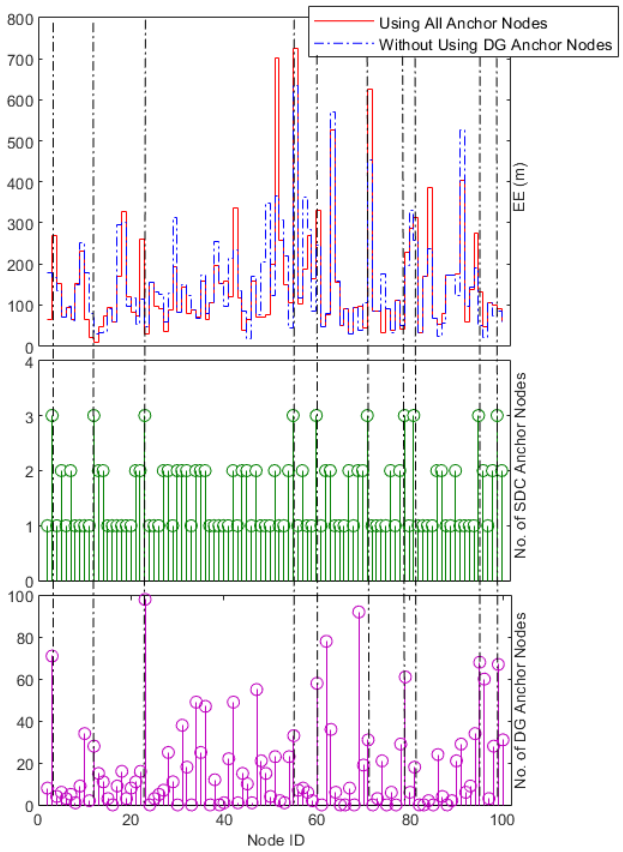


FIGURE 14. Comparing EE values and the number of Anchor nodes.

for 15 out of these 20 DG nodes, which brings the average EE value for Experiment Set V to just 116.60 m.

With an average EE value of 116.60 m in Experiment Set V, there are 1% DG nodes with EE value ≤25 m, 17% DG nodes with EE value ≤50 m, 35% DG nodes with EE value ≤75 m, 56% DG nodes with EE value ≤100 m, 62% DG nodes with EE value ≤116.60 m, 70% DG nodes with EE value ≤150 m and 91% DG nodes with EE value ≤200 m. It is worth mentioning that these EE values were obtained based on the topology data collected with a radio range of 200 m. As discussed in Section VI-A2, we have chosen this radio range value based on the assumption that the DG nodes are mobile zebras. In cases, where the DG nodes represent some other types of animals (e.g. tortoises), a shorter radio range (e.g. 50 m) should be used, in which case, the EE values can be further reduced (e.g. ≤50 m).

VII. EVALUATIONS AND COMPARISONS

In this section, we evaluate and compare the LMM system with related work. This is done in two parts. In part A - Comparison with ZebraNet, we compare the physical characteristics of SDC/DG nodes used in the LMM system with those used in the ZebraNet system [6]. In part B - Comparison with DV-hop Algorithm, we compare the accuracy of the movement traces estimated using our LE method against those estimated using the DV-hop algorithm [17].

A. COMPARISON WITH ZebraNet

This comparison between the LMM system and the ZebraNet system is based on the characteristics of nodes used in the respective systems. The node characteristics used are node weight, node storage capacity, energy cost, and manufacturing and deployment cost. The comparison will be on per node basis. For simplicity, hereafter, a SDC/DG node used in the LMM system is referred to as a LMM node and a node in the ZebraNet system as a ZebraNet node.

1) NODE WEIGHT

Table 6 gives the weights of components installed on each node used in the respective systems. From the table, it can be seen that the total weight of a LMM node is 77 grams, as against 1,151 grams for a ZebraNet node. In other words, a LMM node only weighs 6.7% of the weight of a ZebraNet node.

The top three heaviest components equipped on a ZebraNet node are the solar array, accounting for 47%, the 14 cells, accounting for 25% and the long-range radio, also accounting for 25% of the total weight of a ZebraNet node.

In contrast, each LMM node only consists of two components, a Tmote Sky wireless module and 2 x AA cells. The Tmote Sky wireless module hosts a microcontroller and, an integrated radio and an antenna, supporting short-range communications. Assuming that rechargeable Ni-MH cells are used to power up the Tmote Sky wireless module, then the total node weight amounts to 77 grams.

2) NODE STORAGE CAPACITY

Each ZebraNet node uses GPS-MS1E 640KB on-board flash RAM to store the GPS locations. In their paper [6], Juang, *et al.* stated that, to store 30 GPS locations per hour, and assuming that each GPS location is expressed in 8 bytes, a ZebraNet node requires a RAM capacity of 240 bytes per hour. This implies that a ZebraNet node, with a 640KB RAM space, can store GPS locations for 110 days. The authors also stated that, by applying a data compression rate of about 36%, the RAM space could store the GPS locations for 300 days.

With regard to the RAM capacity requirement on a LMM node, our study (Study-1) shows that, for the given network setting, each node requires just 42 bytes per message. A Tmote Sky 48KB on-board flash RAM is used on each LMM node and this capacity is only 7.5% of the flash RAM capacity used in a ZebraNet node. During Study-1, we obtained 30 TD_{MDC} datasets in a simulation run of 1120 seconds (i.e. 0.31 hour). Assuming that each LMM node contributes data to each TD_{MDC} dataset, then each LMM node can generate 96 ATL messages per hour, with the payload length for each message being 42 bytes. If no data aggregation were used (i.e. at each node, these ATL messages are stored separately), each LMM node would require a little over 4K byte RAM space per hour, thus being able to

TABLE 6. Comparing node weight.

	Component	Weight
ZebraNet	uBlox GPS-MS1E Single-chip GPS/CPU (includes 20MHz Hitachi SH1 32-bit microprocessor (1MB Flash with 640KB for User Data), 12-channel GPS receiver)	8 grams
	Linx SC-PA Short-range Radio (100m, 19.2Kbps)	20 grams
	Long-range Radio (8km, 2.4Kbps) with Packet Modem	296 grams
	14 Sony Lithium-Ion Polymer Cells: (UP503759 AH) 3.7V, 1AH cells	287 grams total
	Solar Array - Unisolar USF5 flexible 5 watt	540 grams
	Total	1,151 grams (2.54 lbs)
LMM	Tmote Sky (includes 8MHz Texas Instruments MSP430 8-bit microcontroller (10KB RAM, 48KB Flash), 250kbps 2.4GHz IEEE 802.15.4 Chipcon Wireless Transceiver, Integrated on-board antenna with 50m range indoors and 125m range outdoors)	23 grams
	2 Rechargeable Nickel-Metal Hydride (Ni-MH) AA Cells: 3.6V, 2300MAH cells	54 grams total
	Total	77 grams (0.17 lbs)

store topology data for about 12 hours. However, with data aggregation, each ATL message actually carries aggregated topology data received from multiple nodes. That is to say, with a message payload length of 42 bytes, an ATL message can carry topology data for a maximum of 24 nodes. During Study-1, we discovered that most of the DG nodes generated just one ATL message each and, on average, each SDC node generated 4 ATL messages, per simulation run. This means that, for most DG nodes, each only requires 42 bytes RAM space and each SDC node, on average, requires 168 bytes RAM space.

3) ENERGY COST

Table 7 gives the energy cost of different operations that are performed by a ZebraNet node and by a LMM node, respectively.

From the table, it can be seen that, for almost all the operations (with an exception of one operation) that are performed by a ZebraNet node, the energy cost incurred is significantly higher than the operations performed by a LMM node. The energy cost for different operations in a ZebraNet node ranges between <1 mA to 1622 mA. The most energy-consuming operation performed by a ZebraNet node is when it uses packet modem and long-range radio to communicate with the BS, and this imposes a cost of 1622 mA. In [6], simulation for data collection is done every two hours and continues over a period of 30 minutes, e.g. data can be collected at the intervals 12:00 – 12:30, 02:00 – 02:30 and so on. During each data collection, each sensor node attempts to discover its neighbours, discover the BS, transfer the collected data to the neighbours and transfer the data to the BS. For this, the authors assume that neighbour and BS discoveries take 30 seconds, whereas the corresponding collected data transfers depend on the available bandwidth and the amount of data to be transferred.

In contrast, for a LMM node, the cost incurred by different operations ranges from 5.1μA to 23 mA, which is just 1.5% of the cost imposed on a ZebraNet node. This significant reduction of 98.5% in the energy cost by the LMM system is due to the fact that it only uses a short-range communication facility to facilitate communications among sensor nodes as well as the communications between sensor nodes and the BSes. For LMM, the simulation for Study-1 is run for 1120 seconds and data is collected every 5 seconds. During each data collection, each DG node sends its topology data to just a 1-hop upstream node. The simulation for Study-2 is done after every 1120 seconds and data is collected just from the SDC nodes. During each such data collection, the MDC node travels to the network site to pull topology data from each of the SDC nodes in a sequential order. This means that for data collections in both Study-1 and Study-2, only short-range communications are used, and in each such data collection a fixed-size message of 42 bytes is sent just to a 1-hop upstream node.

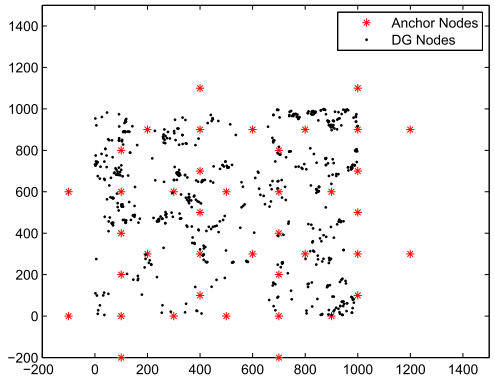
4) MANUFACTURING AND DEPLOYMENT COST

With regard to the manufacturing cost, a ZebraNet node is custom-made, built using several off-the-shelf components and is specifically designed for the ZebraNet system. The individual cost breakdown for these off-the-shelf components is as follows: uBlox GPS-MS1E cost ranges between £36.27 to £62.00 [35], Linx SC-PA short-range radio costs ranges between $36.70 to $46.30 [36], long-range radio with 8km range costs $28.47 [37], Unisolar USF5 solar array costs $85.00 [38] and each of the 14 Lithium-Ion polymer cells cost ranges between £4.20 to £5.10 [39], i.e. total cost ranges between £58.80 to £71.40. This means that the cost of a ZebraNet node can range anywhere between £216.12 to £262.19. On the other hand, a LMM node can be a low-cost ordinary sensor node, e.g. one that is designed for a diverse range of applications and built commercially. Such sensor nodes are usually cheaper than those that are custom-made, e.g. a CM5000 TelosB mote costs €90 (£79.59) [40].

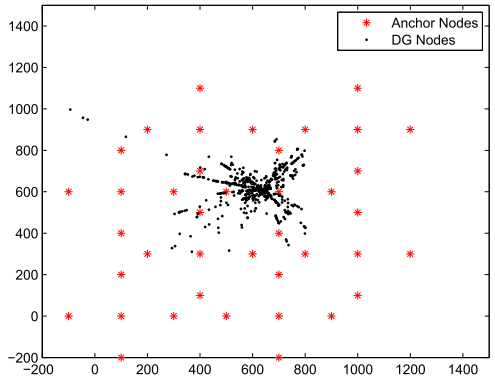
With respect to the deployment cost, to collect data from 'n' nodes, the ZebraNet system will need 'n' nodes but the LMM system will need 'n + m' nodes, where 'm' is the

TABLE 7. Comparing energy cost.

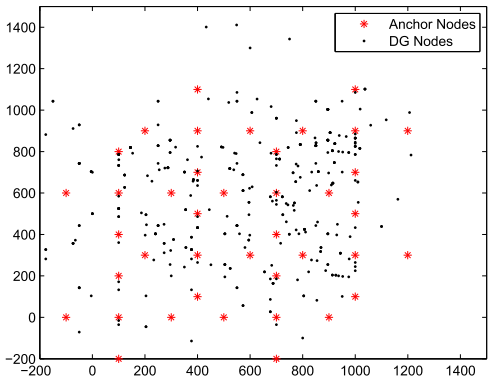
	Collar State	Device and Mode	Current Drain of 3.6V supply
ZebraNet	Stand-by	All	<1mA
	Location Sampling and Storage	GPS-MS1E, Active Antenna	177mA
	Peer Discovery/Transfer Only	GPS-MS1E, Short-range	177mA
	Base Discovery Only	GPS-MS1E, Long-range	432mA
	Simultaneous Peer and Base Search	GPS-MS1E, Short-range + Long-range	469mA
	Transmitting Data to Base	GPS-MS1E, Long-range	1622mA
LMM	Stand-by	All	5.1 µA - 21 µA
	Topology Data Aggregation and Storage	All, Active microcontroller	1800 µA - 2400 µA
	Peer Discovery/Transfer Only	All, Short-range	19.5mA - 23mA
	Base Discovery Only	SDC node, Short-range	19.5mA - 23mA
	Simultaneous Peer and Base Search	SDC node, Short-range	19.5mA - 23mA
	Transmitting Data to Base	SDC node, Short-range	19.5mA - 23mA



(a) Real Movement Traces



(b) Estimated Movement Traces Using DV-hop Algorithm



(c) Estimated Movement Traces Using LE Method

FIGURE 15. Real and estimated movement traces for DG nodes.

number of SDC nodes, which is determined using equation (6). Assuming that the monetary cost per node for the ZebraNet system is 'x' and the monetary cost per node for the LMM system is 'y', then the total deployment cost for the ZebraNet system will be (n×x) and the total deployment costs for the LMM system will be (n×y + m×y).

B. COMPARISON WITH THE DV-HOP ALGORITHM

The LE method is based on the DV-hop algorithm [17], but it differs from the DV-hop algorithm in that the former makes use of dynamic data that are acquired in real-time. This section compares the performance of the LE method with the DV-hop algorithm. Figure 15 contains three sub-figures: Figure 15(a) shows the real movement traces of the DG nodes, Figure 15(b) shows the movement traces estimated using the DV-hop algorithm and Figure 15(c) shows the movement traces estimated using the LE method. The black dots in the figures represent the movement traces of 100 DG nodes, e.g. in Figure 15(c) they are the 1,757 physical locations estimated for the 100 DG nodes. With regard to Figure 15(c), it should

be mentioned that the anchor nodes shown in the figure refer to SDC nodes only, but the movement traces are estimated using both DG and SDC nodes as anchor nodes.

Comparing the results shown in Figure 15(b) with those of Figure 15(a), it can be seen that the DG nodes' movement traces estimated using the DV-hop algorithm (i.e. the black dots in Figure 15(b)) are markedly different from the real movement traces (the black dots shown in Figure 15(a)). The black dots in Figure 15(b) are mostly concentrated in the central part of the network, whereas those in Figure 15(a) are rather spread out across the entire network. This means that the estimation given by the DV-hop algorithm is not a true representation of the nodes' real movement traces. This is because the DV-hop algorithm is designed for an isotropic network (i.e. a network with the same properties in all directions).[4] For a network that consists of DG nodes representing mobile zebras, it is an anisotropic network,[5] and for an anisotropic network, the DV-hop algorithm cannot produce accurate estimations of DG nodes' physical locations.

In contrast, the results produced by our LE method, as shown in Figure 15(c), are a much closer reflection of the movement traces of the DG nodes shown in Figure 15(a). This indicates that the LE method can estimate the movement traces of the DG nodes much more accurately. This is due to two reasons. Firstly, the DG nodes' hop distances used in the LE method are acquired in real-time and they are more accurate than those used in the DV-hop algorithm, where the hop distances are calculated based on just the shortest path algorithm. Secondly, the anchor nodes used in the DV-hop algorithm are pre-defined nodes (i.e. just the stationary SDC nodes), whereas in the LE method, the anchor nodes are selected dynamically and consist of both SDC and DG nodes.

Figure 16 compares the EE values of the estimation results produced by the DV-hop algorithm and the LE method, respectively. The two horizontal lines are, respectively, the average EE values of the two sets of results. The average EE value produced by the DV-hop algorithm is 303.3 m, in contrast to 116.60 m by the LE method. This means that, with the LE method, the average EE value is reduced by 61.56% as compared to the DV-hop algorithm. Of the 100 DG nodes, only 8 DG nodes have a lower EE value when using the DV-hop algorithm as compared to the LE method. It is interesting to note that 3 out of these 8 DG nodes (i.e. DG nodes 17, 18, 24, 33, 46, 63, 84, 90) have already been identified (in our study presented in Section VI-B2) as the DG nodes that did not move themselves but their parent DG node, carrying their topology data, moved. For the remaining 92 DG nodes, the LE method produces lower

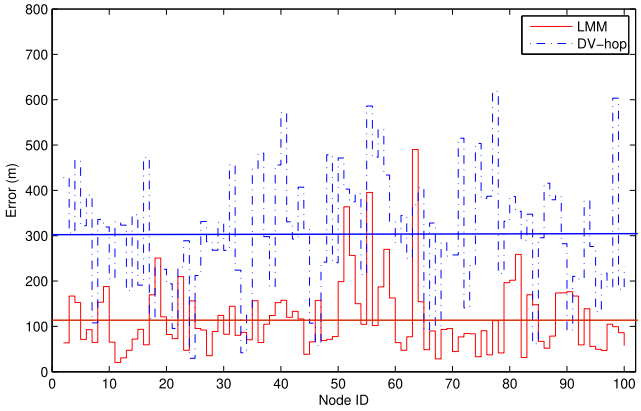


FIGURE 16. Comparing EE values in the estimated physical locations.

EE values, i.e. produces more accurate estimations, than the DV-hop algorithm.

VIII. CONCLUSION

This paper has presented the design and evaluation of a novel system, the LMM system, for wildlife tracking. The LMM system consists of a multi-zone multi-hierarchy communication structure, three novel protocols to facilitate topology data collection, and a novel method for the estimation of DG nodes' movement traces based on the collected topology data. We have investigated the applicability and effectiveness of the LMM system for tracking zebras using real-life datasets and two simulation studies. In the first study, we experimented with various parameter value settings and studied their impacts on the effectiveness of the topology data collections. The results of this study have been used for the design of the LMM system and the selection of parameter values for the deployment of the system. Then, in the second study, we evaluated the accuracy performance of the system by applying it to the estimation of zebras' movement traces and compared the results against their real movement traces. We also analysed the storage requirements, and energy and deployment cost of the LMM system and compared them with those of the most related systems. The evaluation results indicate that the LMM system only costs tiny fractions of the costs required by the other systems, but provides more accurate wildlife tracking results.

The results and the findings demonstrate that, by making use of an appropriate communication structure, data structure based information encoding and data aggregation, and short-range peer-to-peer communications, we can achieve cost-efficient real-time data collections even with very resource-constrained sensor nodes and that, by making use of the collected data properly, we can estimate (learn) other useful information, such as the movement traces of moving animals.

[4]In an isotropic network, the shortest path between any two nodes can be represented by a straight line and the nodes' distances in hops can be accurately converted into their distances in meters.

[5]When the network is anisotropic, the shortest path between two nodes may form a curve instead of a straight line. With this curve, the nodes' distances in hops would not be accurately converted into their distances in meters.

REFERENCES

[1] P. Wamuyu, "A conceptual framework for implementing a WSN based cattle recovery system in case of cattle rustling in kenya," *Technologies*, vol. 5, no. 3, p. 54, Aug. 2017.

[2] M. Magno, F. Vultier, B. Szebedy, and L. Benini, "Long-term monitoring of small-sized birds using a miniaturized bluetooth-low-energy sensor node," in *Proc. IEEE Sensors*, Oct. 2017, pp. 1–3.

[3] L. Bracciale, A. Catini, G. Gentile, and P. Loreti, "Delay tolerant wireless sensor network for animal monitoring: The pink iguana case," in *Proc. Int. Conf. Appl. Electron. Pervading Ind., Environ. Soc.* Cham, Switzerland: Springer, Sep. 2016, pp. 18–26.

[4] I. E. Radoi, J. Mann, and D. K. Arvind, "Tracking and monitoring horses in the wild using wireless sensor networks," in *Proc. IEEE 11th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2015, pp. 732–739.

[5] E. S. Nadimi, R. N. Jørgensen, V. Blanes-Vidal, and S. Christensen, "Monitoring and classifying animal behavior using ZigBee-based mobile ad hoc wireless sensor networks and artificial neural networks," *Comput. Electron. Agricult.*, vol. 82, pp. 44–54, Mar. 2012.

[6] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," *ACM SIGARCH Comput. Archit. News*, vol. 30, no. 5, pp. 96–107, Dec. 2002.

[7] P. Zhang, C. M. Sadler, S. A. Lyon, and M. Martonosi, "Hardware design experiences in ZebraNet," in *Proc. 2nd Int. Conf. Embedded Networked Sensor Syst. SenSys*, 2004, pp. 227–238.

[8] J. Xu, G. Solmaz, R. Rahmatizadeh, D. Turgut, and L. Boloni, "Animal monitoring with unmanned aerial vehicle-aided wireless sensor networks," in *Proc. IEEE 40th Conf. Local Comput. Netw. (LCN)*, Oct. 2015, pp. 125–132.

[9] J. Xu, G. Solmaz, R. Rahmatizadeh, D. Turgut, and L. Boloni, "Internet of Things applications: Animal monitoring with unmanned aerial vehicle," 2016, *arXiv:1610.05287*. [Online]. Available: https://arxiv.org/abs/1610.05287

[10] G. L. Kirkland, "Guidelines for the capture, handling, and care of mammals as approved by the American society of mammalogists," *J. Mammalogy*, vol. 79, no. 4, pp. 1416–1431, Dec. 1998.

[11] Resources Inventory Committee, "Wildlife radio-telemetry: Standards for components of British Columbia's biodiversity. No. 5, version 2.0," Ministry Environment, Lands Parks, Resour. Inventory Br., Victoria, BC, Canada, Tech. Rep., 1998.

[12] D. W. Macdonald, "Radio-tracking: Some applications and limitations," in *Animal Marking*. London, U.K.: Springer, 1978, pp. 192–204.

[13] C. Brooks, C. Bonyongo, and S. Harris, "Effects of global positioning system collar weight on zebra behavior and location error," *J. Wildlife Manage.*, vol. 72, no. 2, pp. 527–534, Feb. 2008.

[14] C. E. Coughlin and Y. van Heezik, "Weighed down by science: Do collar-mounted devices affect domestic cat Behaviour and movement," *Wildlife Res.*, vol. 41, no. 7, pp. 606–614, 2015.

[15] M. Corporation. (2006). *Datasheet, Tmote Sky*. Accessed: Feb. 11, 2018. [Online]. Available: http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-s%ky-datasheet.pdf/

[16] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "Tinyos: An operating system for sensor networks," in *Ambient Intelligence*. Berlin, Germany: Springer, 2005, pp. 115–148.

[17] D. Niculescu and B. Nath, "DV based positioning in Ad Hoc networks," *Telecommun. Syst.*, vol. 22, nos. 1–4, pp. 267–280, 2003.

[18] Y. Wang, P. Zhang, T. Liu, C. Sadler, and M. Martonosi. (Feb. 2007). *CRAWDAD Dataset Princeton/Zebranet (v. 2007-02-14)*. Downloaded From. [Online]. Available: https://crawdad.org/princeton/zebranet/20070214

[19] H. Gao, C. Liu, Y. Li, and X. Yang, "V2 VR: Reliable hybrid-network-oriented V2 V data transmission and routing considering RSUs and connectivity probability," *IEEE Trans. Intell. Transp. Syst.*, early access, Apr. 13, 2020, doi: 10.1109/TITS.2020.2983835.

[20] H. Gao, W. Huang, and Q. Zou, "A dynamic planning framework for qos-based mobile service composition under cloud-edge hybrid environments," in *Proc. Int. Conf. Collaborative Comput., Netw., Appl. Worksharing*, vol. 292. London, U.K.: Springer, Aug. 2019, p. 58.

[21] R. D. Taber and I. M. Cowan, "Capturing and marking wild animals," in *Wildlife Management Techniques*. Washington, DC, USA: Wildlife Society, 1969, pp. 277–317.

[22] T. Nietfeld, "Wildlife marking techniques," in *Research and Management Techniques for Wildlife and Habitats*. Bethesda, MD, USA: The Wildlife Society, 1994, pp. 140–168.

[23] D. L. Murray and M. R. Fuller, "A critical review of the effects of marking on the biology of vertebrates," in *Research Techniques in Animal Ecology, Controversies and Consequences*. New York, NY, USA: Columbia Univ. Press, 2000, pp. 15–64.

[24] O. Gnawali, R. Fonseca, M. Kazandjieva, D. Moss, and P. Alexander, "CTP: An efficient, robust, and reliable collection tree protocol for wireless sensor networks," *ACM Trans. Sensor Netw.*, vol. 10, no. 1, p. 16, 2013.

[25] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis, "Collection tree protocol," in *Proc. 7th ACM Conf. Embedded Netw. Sensor Syst.*, 2009, pp. 1–14.

[26] R. Fonseca, O. Gnawali, K. Jamieson, S. Kim, P. Levis, and A. Woo, "The collection tree protocol (CTP)," *TinyOS TEP*, vol. 123, p. 2, Aug. 2006.

[27] Memsic. (2006). *Datasheet, MICAz*. Accessed: Jul. 9, 2018. [Online]. Available: http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.% pdf/

[28] G. Anastasi, A. Falchi, A. Passarella, M. Conti, and E. Gregori, "Performance measurements of motes sensor networks," in *Proc. 7th ACM Int. Symp. Modeling, Anal. Simulation wireless Mobile Syst. MSWiM*, 2004, pp. 174–181.

[29] *Zebra Facts—Animal Facts Encyclopedia*. Accessed: Mar. 30, 2020. [Online]. Available: https://www.animalfactsencyclopedia.com/Zebra-facts.html

[30] A. Naureen, N. Zhang, and S. Furber, "Identifying energy holes in randomly deployed hierarchical wireless sensor networks," *IEEE Access*, vol. 5, pp. 21395–21418, 2017.

[31] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with COOJA," in *Proc. 31st IEEE Conf. Local Comput. Netw.*, Nov. 2006, pp. 641–648.

[32] D. Gay, P. Levis, R. Von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesc language: A holistic approach to networked embedded systems," *ACM Sigplan Notices*, vol. 49, no. 4, pp. 41–51, 2003.

[33] K. Nitesh and P. K. Jana, "Relay node placement with assured coverage and connectivity: A jarvis march approach," *Wireless Pers. Commun.*, vol. 98, no. 1, pp. 1361–1381, Jan. 2018.

[34] *MATLAB Code for DV-Hop*. Accessed: Mar. 15, 2018. [Online]. Available: http://en.pudn.com/Download/item/id/702307.html

[35] *GPS-MS1E u-Blox CAD Model Download | Octapart*. Accessed: May 9, 2020. [Online]. Available: https://octopart.com/gps-ms1e-u-blox-30335590

[36] *Linx RF Modules*. Accessed: May 9, 2020. [Online]. Available: https://user.eng.umd.edu/~neil/TRX/Datasheets/pricelist_linx.pdf

[37] *E90-DTU-170l30 Wireless Transceiver RS232 RS485 170MHz LoRa 1W Long Range 8km RF Module Radio Modem LoRa for Data Transmission*. Accessed: May 9, 2020. [Online]. Available: https://www.aliexpress.com/i/32857798817.html

[38] *Uni-Solar Solar Panels/Modules*. Accessed: May 9, 2020. [Online]. Available: http://www.innovationhouse.com/products/unisolar.html

[39] *Polymer Lithium Ion Battery (3.7v, 1ah)*. Accessed: May 9, 2020. [Online]. Available: https://www.kitronik.co.uk/4652-polymer-lithium-ion-battery-1ah.html

[40] (2011). *Datasheet, Mtm-cm5000-Msp Mote*. Accessed: Jul. 4, 2018. [Online]. Available: https://www.advanticsys.com/shop/mtmcm5000msp-p-14.html/

AYESHA NAUREEN received the B.Eng. degree in software engineering and the M.S. degree in information security from the National University of Sciences and Technology, Pakistan, in 2005 and 2008, respectively, and the Ph.D. degree in computer science from The University of Manchester, U.K., in 2019.

From 2008 to 2014, she was a Lecturer with the National University of Sciences and Technology, Pakistan. Her most recent work is in the design and application of blockchain technologies for decentralized energy markets. Her research interests include security, privacy, blockchain, protocol design, and performance analysis in wireless sensor networks.

NING ZHANG received the B.Sc. degree (Hons.) in electronic engineering from Dalian Maritime University, China, and the Ph.D. degree in electronic engineering from the University of Kent, Canterbury. Since 2000, she has been with the Department of Computer Science, The University of Manchester, U.K., where she is currently a Senior Lecturer. Her research interests include security and privacy in networked and distributed systems, such as ubiquitous computing, electronic commerce, wireless sensor networks, and cloud computing with a focus on security protocol designs, risk-based authentication and access control, and trust management.

STEVE FURBER (Fellow, IEEE) received the B.A. degree in mathematics and the Ph.D. degree in aerodynamics from the University of Cambridge, U.K. He spent the 1980s at Acorn Computers, where he was a Principal Designer of the BBC Microcomputer and the ARM 32-bit RISC microprocessor. Over 130 billion variants of the ARM processor have since been manufactured, powering much of the world's mobile and embedded computing. He moved to the ICL Chair at Manchester, in 1990, where he leads research into asynchronous and low-power systems and, more recently, neural systems engineering, where the SpiNNaker project has delivered a computer incorporating a million ARM processors optimized for brain modeling applications. He is currently an ICL Professor of computer engineering with the Department of Computer Science, The University of Manchester, U.K. He is a CBE, a Fellow of the Royal Society, and a Fellow of the Royal Academy of Engineering.

QI SHI received the Ph.D. degree in computing from the Dalian University of Technology, China. He worked as a Research Associate with the University of York, U.K. He joined Liverpool John Moores University (LJMU), U.K., as a Lecturer. He was a Reader with LJMU, where he was also a Professor. He is currently a Professor of computer security and the Director of the PROTECT Research Centre, Department of Computer Science, LJMU. He has many years of research experience in a number of areas, such as the IoT security, intrusion detection, secure service composition, privacy-preserving data aggregation, cryptography, computer forensics, formal security models, and cloud security. He has also played a key role in many funded research and development projects related to his research topics. He has published over 250 articles in international conference proceedings and journals. He served in a number of conference IPCs and journal editorial boards.